Datenverarbeitung

Nachdem die Datengrundlage geschaffen wurde, müssen die ankommenden Daten für die weitere Verwendung verarbeitet werden. Falls möglich sollte dafür zunächst eine explorative Datenanalyse durchgeführt werden. Die eigentliche Datenverarbeitung lässt sich in zwei Kategorien einteilen: Echtzeit-Verarbeitung und asynchrone Verarbeitung.

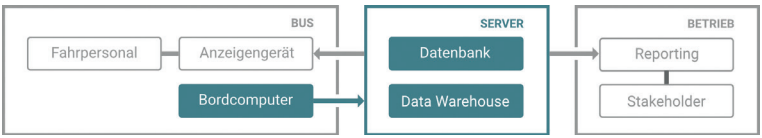

Abbildung 3: Übersichtsgrafik – Datenverarbeitung

Explorative Datenanalyse

Falls es zu Beginn dieser Phase bereits eine Datengrundlage gibt, sollte zunächst eine explorative Datenanalyse durchgeführt werden. Deren Ziel ist es, einen Eindruck von der Struktur und Qualität der Daten in Bezug auf Vollständigkeit und Korrektheit zu bekommen. Die daraus gewonnenen Erkenntnisse sollten in einem zentralen Dokument (im Folgenden „Logbuch" genannt) festgehalten werden. In diesem sollten möglichst alle, aber zumindest alle relevanten Kanäle der Rohdaten beschrieben werden. Dazu gehören: eine Verbindung des Namens des Kanals zu einem verständlichen Namen, eine plausible Reichweite von Werten, die der Kanal annehmen kann und eine optionale Beschreibung des Kanals.

Zunächst muss ein geeignetes Tool zur Untersuchung der Daten gewählt werden. Das wohl bekannteste Tool zur Datenauswertung ist Microsoft Excel. Alternativ existieren Programmiersprachen, die sich vollständig (R) oder teilweise mit bestimmten Libraries (Python mit Pandas) auf Datenauswertung spezialisiert haben. Diese Programmiersprachen haben gegenüber Excel einige Vorteile: Sie sind in der Lage auch größere Datenmengen zu verarbeiten, Datenauswertungen sind besser reproduzierbar und der entstehende Code kann eventuell wiederverwendet werden. Der Nachteil ist ein erhöhter Aufwand in der Einarbeitung, dieser kann sich mittel- und langfristig aber lohnen. Bei der Wahl des Tools sollte vor allem das Vorwissen in der Arbeitsgruppe berücksichtigt werden.

Echtzeitverarbeitung

Das Ziel der Echtzeitverarbeitung ist es, die Daten mit möglichst wenig Latenz und möglichst hoher Qualität vom Bordcomputer auf das Tablet zu übermitteln. Wie oben beschrieben lässt sich ein Umweg über einen Webserver nicht immer vermeiden und ist für eine zentrale Überwachung notwendig. Zunächst müssen die Rohdaten für eine spätere Verarbeitung gespeichert werden. Außerdem müssen die Rohdaten validiert und dann abgeglichen werden. Grundsätzlich lässt sich jede für den Server geeignete Programmiersprache nutzen. Solange die Performance kein Problem darstellt, kann die gleiche Programmiersprache wie in der explorativen Datenauswertung verwendet werden, wodurch sich möglicherweise Code wiederverwenden lässt.

Speicherung

Nach der Übertragung muss sichergestellt werden, dass die übertragenen Daten gespeichert werden. Die Wahl der Datenbank sollte hier vor allem von der Skalierung des Systems abhängen. Im Folgenden wird davon ausgegangen, dass kein verteiltes System zur Speicherung der Daten nötig ist. Dann kann eine relationale Datenbank wie MySQL oder PostgreSQL verwendet werden. Die Datenbank für die Speicherung der Daten sollte auf das Schreiben und nicht auf das Abrufen der Daten ausgelegt sein. Zur Indexierung der Daten bietet sich der Timestamp der Roh-

daten an. Außerdem ist es sinnvoll einen Timestamp beim Empfang der Daten zu speichern, um die Latenz zwischen Bordcomputer und Webserver beobachten zu können.

Validierung der Fahrdaten

Bei der Validierung der Fahrdaten sollte auf Erkenntnisse aus der explorativen Datenanalyse und das dort entstandene Logbuch zurückgegriffen werden. Viele der Datenkanäle können nur Werte aus einem begrenzten Wertebereich annehmen. Beispielsweise kann der Ladestand der Batterie nicht über ihre maximale Kapazität steigen und die gefahrenen Kilometer nicht negativ sein. Dieses Wissen sollte so gut wie möglich genutzt werden, um Fehler frühzeitig zu erkennen und letztendliche Fehler in der Anzeige zu vermeiden. Weiterhin sollten die ankommenden Daten auf ihre Reihenfolge geprüft werden. Dies kann mithilfe des Logging Timestamps realisiert werden. Letztlich sollte das System einen Ausfall des Loggings erkennen und Client-Systemen entsprechende Informationen bereitstellen, damit diese angemessen reagieren können. Falls möglich, sollte ein automatisches Reporting an die für den Bordcomputer Verantwortlichen Personen stattfinden, damit das Problem schnellstmöglich behoben werden kann.

Aggregierung und Augmentierung der Fahrdaten

Nach der Validierung der Fahrdaten können die Rohdaten aggregiert und augmentiert werden. Grundsätzlich ist dies sowohl auf dem Server und dem Client möglich. Beide Möglichkeiten bieten Vor- und Nachteile:

Die Aggregierung auf dem Server ist vor allem dann sinnvoll, wenn mehrere Client Systeme diese Daten benötigen. Damit muss die Verarbeitung nur einmal stattfinden. Auch sehr rechenintensive Verarbeitungen sollten – wegen der leistungsstärkeren Hardware – auf dem Server durchgeführt werden.

Ein Nachteil der Aggregierung und Augmentierung auf dem Server ist, dass solch ein *State* auf dem Server gehalten werden muss. Solange das System nicht verteilt ist sondern auf einem einzelnen Server basiert, stellt dies kein Problem dar. Muss das System später auf mehrere Server verteilt werden, kann dies allerdings zu Problemen führen. Falls möglich ist also eine Verarbeitung, die einen *State* auf dem Server nötig macht, zu vermeiden.

Inhaltlich gibt es viele Möglichkeiten die Rohdaten zu erweitern. Inwiefern das sinnvoll ist, hängt von den Zielen der Clientsysteme ab. Zwei sinnvolle Beispiele sind: Die Verarbeitung von GPS-Daten um Haltestellen zu erkennen und die Aggregierung vom Verbrauch über eine zurückliegende Strecke / Zeit.

Verfügbarmachung durch eventbasiertes Publishing

Für die weitere Verwertung der Echtzeitdaten müssen die nun verarbeiteten Daten für Client-Systeme zur Verfügung gestellt werden. Hierfür bietet sich eine eventbasierte Architektur an. Hierbei können einzelne Event Streams (also z. B. der Datenstrom eines Bordcomputers) von Client-Systemen abonniert werden. Somit kann pro Bus ein Event Stream erstellt werden, wobei die sich jeweils in dem Bus befindende App nur diesen Stream zu abonnieren braucht. Eine zentrale App, wie z. B. eine Übersicht der Ladestände aller Busse kann wiederum alle Streams abonnieren. Auch das Erweitern der Systemarchitektur durch neue Anwendungen wird so ermöglicht. Für Systeme, die nicht mehr als einen Server benötigen, bietet sich Redis als Lösung an. Muss das System auf mehrere Server verteilt werden, kann Apache Kafka eine geeignete Wahl sein.

Asynchrone Datenverarbeitung

Unter die asynchrone Datenverarbeitung fallen alle Verarbeitungsschritte, die nicht in Echtzeit geschehen. Diese können einmalig oder regelmäßig in unterschiedlichen Zeitabständen stattfinden.

Data Warehouse

Ein Data Warehouse ist eine Datenbank, die Daten in einer für Anfragen spezialisierten Form bereitstellt. Hierfür werden die Daten mithilfe von Batch Jobs verarbeitet. Batch Jobs sind automatisierte Datenverarbeitungsprogramme, die regelmäßig zu bestimmten Zeitpunkten stattfinden. Das Ziel ist es, die Daten in geeigneter Form zu aggregieren und augmentieren, um Anfragen zu beschleunigen.

Wie genau die Daten aggregiert werden, ist abhängig von den Anforderungen. Einige sinnvolle Aggregierungen sind beispielsweise:

- Aggregation einzelner Linienfahrten
- Aggregation von Umläufen einzelner Busse
- Verbrauch auf bestimmten Linien/Linienabschnitten
- Wöchentliche/Monatliche/Quartalsmäßige Analysen

Das Ziel eines solchen Data Warehouses ist die bessere Verfügbarkeit der Daten. Diese können dann effizient von anderen Systemen, wie z. B. einem Reporting Tool genutzt werden.

Prädiktive Modelle

Mit Hilfe von Machine Learning ist es möglich auf Basis der Daten prädiktive Modelle zu erstellen. Gerade im Bus-Betrieb bieten sich hier durch die immer gleichen Strecken Möglichkeiten, die nicht bei allen Elektrofahrzeugen gegeben sind. Ein Beispiel für eine sinnvolle Prädiktion ist beispielsweise eine Vorhersage, mit welcher Akkuladung ein Bus eine Strecke beendet. Hierfür geeignete Libraries sind z. B. Scitkit-Learn (Python), Pytorch (Python) und Keras / Tensorflow (R und Python).

Zusammenfassung & Checkliste/Prozess

Bevor die Rohdaten – wie im nächsten Kapitel beschrieben – in Assistenzsystemen verwertet werden können, müssen sie vorverarbeitet werden. Dabei kann zwischen der Vorbereitung für Echtzeit- und asynchro-

nen Anwendungen unterschieden werden. Stellen Sie sicher, dass Sie für Echtzeitanwendungen jeden Punkt der folgenden Checkliste abhaken können.

- ☐ **Es wurde ein Logbuch angelegt und für relevante Personen verfügbar gemacht**,
  d. h. es existiert ein Dokument, z. B. in Form einer Tabelle, in dem Informationen über Datenkanäle festgehalten werden.

- ☐ **Alle Daten werden nach der Übertragung zuverlässig gespeichert**,
  d. h. es wurde eine geeignete Datenbank gewählt und im Rahmen der Echtzeitverarbeitung die Speicherung realisiert.

- ☐ **Alle Daten werden vor der Veröffentlichung validiert**,
  d. h. es ist sichergestellt, dass Daten korrekt und in der richtigen Reihenfolge dargestellt werden.

- ☐ **Echtzeit-Daten werden mit einer eventbasierten Technologie veröffentlicht**,
  d. h. es wurde eine geeignete Technologie gewählt und in die Echtzeitverarbeitung integriert.

Für weitere Anwendungen sollten zudem die folgenden Punkte bestätigt werden können:

- ☐ **Eine explorative Datenanalyse wurde durchgeführt und die Ergebnisse im Logbuch festgehalten**,
  d. h. die Daten wurden grundlegend auf ihre Qualität und Besonderheiten untersucht und die Ergebnisse im Logbuch festgehalten.

- ☐ **Es wurden Skripte erstellt und ihre automatisierte Ausführung ermöglicht**,
  d. h. Prädiktionsmodelle können trainiert und ein potenzielles Data Warehouse gefüllt werden.

- ☐ **Es wurde eine API für das Data Warehouse entwickelt**,
  d. h. Client Systeme können die Daten des Data Warehouses effizient nutzen.