# ANALYSE UND KONZEPTIONIERUNG EINES VERTEILTEN SYSTEMS ZUR KOMMUNIKATION VON BETRIEBSDATEN UND STATISTIKEN IN INTRALOGISTIKANLAGEN

Thomas Zimmermann B.Sc. und Professor Dr. Frank Herrmann

Innovationszentrum für Produktionslogistik und Fabrikplanung
Ostbayerische Technische Hochschule Regensburg (OTH Regensburg)
Prüfeninger Str. 58, 93049 Regensburg, Germany

**SCHLÜSSELWÖRTER**

Intralogistik, Betriebsdaten, Zeitreihen, verteilte Systeme, Modellierung, Anforderungsanalyse

**ABSTRACT**

Im vorliegenden Artikel werden Vorarbeiten zur Entwicklung eines Betriebsdatenverwaltungssystems für Intralogistikanlagen des Herstellers TGW Software Services beschrieben. Das beschriebene Vorgehen umfasst dabei im Wesentlichen vier Schritte.

Zunächst wird der aktuelle Stand in der Software des betrachteten Unternehmens hinsichtlich des Sammelns, Speicherns und Auswertens von Betriebsdaten analysiert.

Daran schließt sich eine Anforderungsanalyse für die Betriebsdatenverwaltung an. Zusammen mit den betroffenen Entwicklungsleitern werden die grundlegenden Ziele und Grenzen der zu entwickelnden Betriebsdatenverwaltung definiert. Es werden Quellen von Anforderungen für die Komponente gesammelt und alle funktionalen und nichtfunktionalen Anforderungen anhand dieser Quellen erarbeitet.

Basierend darauf wird ein erstes Modell erstellt, das die Entitäten, Beziehungen und Abläufe, die sich aus den ermittelten funktionalen Anforderungen ergeben, in konsolidierter Form zusammenfasst. Dieses wird zusammen mit einigen Überlegungen, die zu diesem Modell geführt haben, beschrieben.

Zuletzt werden dann verschiedene mögliche Konzepte zum grundlegenden technischen Aufbau einer Implementierung der Betriebsdatenverwaltung diskutiert, um schließlich anhand der definierten Qualitätsziele eine Empfehlung für einen Konzeptvorschlag abzugeben.

## EINFÜHRUNG

Diese Arbeit wurde in Zusammenarbeit mit der Firma TGW Software Services (kurz *TSS*) erstellt, die zur *TGW Logistics Group* (kurz *TGW*) gehört.

Der Fokus der TSS liegt unter anderem auf dem Einsatz und der Weiterentwicklung der modularen, standardisierten Softwareplattform *iWACS®* (kurz für *integrated Warehouse Administration and Control System*), die alle für eine moderne Intralogistikanlage benötigten Softwarefunktionalitäten bietet. iWACS® besteht aus mehreren Modulen, die unterschiedliche Aufgaben abdecken und für jedes Intralogistikprojekt nach Bedarf kombiniert werden. Alle Module sind nach dem Client-Server-Prinzip aufgebaut, wobei der Server üblicherweise im Hintergrund auf eine relationale Datenbank zugreift, in der er alle für den Betrieb notwendigen Daten ablegt.

Im Kontext von TGW-Intralogistikprojekten werden einige typische Begriffe verwendet, die hier kurz vorgestellt werden.

Der Begriff *Anlage* bezeichnet im Folgenden immer ein vollständiges Intralogistiksystem, das sich an einem bestimmten Standort befindet. Synonym wird – aus Sicht der TGW als Realisierer von Intralogistikanlagen – teilweise auch der Begriff *Projekt* verwendet.

Eine *TE* (kurz für *Transporteinheit*) ist ein Hilfsmittel, mit dem Ware in einer Anlage bewegt wird. Typische TEs sind Paletten, Behälter oder Kartons. Eine TE ist immer mit einer eindeutigen Nummer (meist in Form eines Barcodes) versehen.

Ein *Arbeitsplatz* ist ein Ort innerhalb einer Logistikanlage, an dem TEs durch Menschen bearbeitet werden. An einem Arbeitsplatz können z. B. Waren einer TE entnommen, geprüft, hinzugefügt oder umgepackt werden. Dies ist beispielsweise für Qualitätskontrollen und zur Kommissionierung notwendig.

### Ausgangssituation

Das Modul **iWACS®.SAV** (*Service, Analysis, Visualization*) dient als Visualisierungssoftware. Es erfragt aktuelle Zustandsdaten von den SPSen der Anlage und zeigt diese dem Benutzer in übersichtlicher Form an. Außerdem ruft das .SAV auch sogenannte Alarme, also Meldungen über Probleme, von den anderen Modulen ab und speichert diese, zusammen mit den SPS-seitigen Störungen, in einem Protokoll.

Ende 2016 wurde in der TSS beschlossen, iWACS®.SAV vollständig neu zu implementieren. Diese Entscheidung gründete auf einer Reihe verschiedener Faktoren:

- Das bisherige .SAV basiert auf z. T. sehr alten Komponenten und Grundstrukturen; dadurch ist es teilweise veraltet und kaum erweiterbar.
- Mit dem bisherigen .SAV sind für jedes Projekt zahlreiche manuelle Einstellungen und Konfigurationen notwendig, die bei der Realisierung viel Zeit kosten.
- Das .SAV soll inzwischen nicht nur für die Projekte der TSS, sondern für alle Intralogistikprojekte in der gesamten TGW-Gruppe eingesetzt werden; dadurch kommen eine Vielzahl neuer Problemstellungen und Anforderungen zustande.
- Eine groß angelegte, konzernweite Anforderungsermittlung hinsichtlich Intralogistik-Visualisierungssoftware mit zahlreichen Stakeholdern ergab über 300 Anforderungen, von denen viele mit dem bisherigen .SAV nicht umgesetzt werden können.
- Kommerziell verfügbare, ähnliche Produkte decken den benötigten Funktionsumfang nicht im geforderten Maße ab.

Insbesondere kamen bei der Anforderungsermittlung viele Punkte auf, aus denen klar wurde, dass in heutigen, durch das allgemeine Wachstum immer komplexer werdenden Intralogistikprojekten eine anlagenweite, zentrale Software zur Präsentation aller wichtigen Daten aus allen Modulen und Systemen benötigt wird.

Ein wesentlicher Bestandteil davon ist die Darstellung von sogenannten Betriebsdaten und Statistiken aller Module und Systeme in der Anlage – dies reicht von Leistungsdaten bestimmter Anlagenteile über den aktuellen Fortschritt im Tagesarbeitspensum und Rechnerantwortzeiten bis hin zur Überwachung von Temperatur und Stromverbrauch einzelner Bauteile. Der Begriff *Betriebsdaten* bezeichnet hierbei im Folgenden immer sämtliche Daten, die vergangene Ereignisse, Vorgänge oder Zustände in einer Logistikanlage quantitativ beschreiben.

Da diese zukünftig in .SAV benötigten Daten durch zahlreiche verschiedene Systeme erzeugt werden, ist es notwendig, diese in ein einheitliches Format zu überführen und klar zu strukturieren. Anschließend müssen die Daten, die z. T. in sehr hohen Frequenzen und damit großen Mengen anfallen, schnell und sicher zwischen Datenquelle und .SAV transportiert werden können. Aus diesem Grund ist es notwendig, eine zentrale Verwaltungskomponente zum Austausch von Betriebsdaten und Statistiken zwischen den einzelnen Systemen in einer Intralogistikanlage zu schaffen, die diese Aufgaben übernimmt.

Auch für die übrigen Module und hinsichtlich einer permanenten Langzeitarchivierung kann eine solche zentrale Komponente nützlich sein. Ein zentrales Datenarchiv könnte z. B. für nachträgliche, großangelegte Auswertungen eingesetzt werden, um Zusammenhänge auf verschiedenen Ebenen der Anlagensteuerung zu erkennen oder um Wartungsintervalle besser bestimmen zu können.

## ISTANALYSE

Da die Sammlung, Speicherung, Darstellung und Auswertung von Betriebsdaten in vielen Anwendungsfällen notwendig ist, wird diese in iWACS® auch bereits an einigen Stellen praktiziert – wenn auch nicht über ein einheitliches System oder Format. Daher werden hier zunächst die bisher verwendeten Lösungen kurz vorgestellt.

### Aufzeichnung von quantifizierbaren Ereignissen mithilfe des Statistikframeworks

Das *Statistikframework* ist eine kleine interne Library, die dazu dient, numerische Werte, die von bestimmten Ereignissen generiert werden, in über einen bestimmten Zeitintervall aufsummierter Form in der Datenbank eines Moduls festzuhalten. Typische Beispiele für solche Ereignisse sind die Bearbeitung einer TE an einem Arbeitsplatz, Ein- und Auslagerungen in einem Lager oder die Dauern bestimmter Verarbeitungsschritte.

Das Modul meldet dabei dem Statistikframework pro Ereignis einen Datensatz, der zum einen Felder enthält, die das Ereignis kategorisieren und zum anderen Felder mit den eigentlichen numerischen Daten dieses Ereignisses. Das Statistikframework summiert die numerischen Daten aller Ereignisse aus einer Kategorie, die in dasselbe (konfigurierbare) Zeitintervall fallen, auf. Diese Datensätze können dann im Client des Moduls angezeigt werden, sodass im Nachhinein – im Rahmen der konfigurierten Genauigkeit – Gesamtsummen und Durchschnitte oder auch Anteile über einen längeren Zeitraum hinweg nachvollzogen werden können.

### Performancedatenprotokollierung mittels RRDB-Logging

Das *RRDB-Logging* ist eine firmeninterne Library, die in verschiedenen iWACS®-Modulen dazu verwendet wird, Betriebsdaten aufzuzeichnen und zu aggregieren; insbesondere für modulinterne Performancemessungen.

Eine aufzuzeichnende Zeitreihe muss vom jeweiligen Modul am RRDB-Logging angemeldet werden. Das RRDB-Logging fragt dann einmal pro Minute alle registrierten Signale ab und speichert die gewonnenen Werte in CSV-Dateien. Im Laufe der Zeit werden dann die gespeicherten Werte nach und nach aggregiert, um Speicherplatz zu sparen. Somit ist sichergestellt, dass die Daten (in der gröbsten Aggregationsstufe) bis zu ein Jahr lang aufbewahrt werden können und zugleich nicht übermäßig viel Speicherplatz benötigt wird.

Da es sich um CSV-Dateien handelt, können die gespeicherten Werte mit relativ geringem Aufwand durch Tabellenkalkulationsprogramme wie Excel dargestellt und ausgewertet werden.

Typische Daten, die mit dem RRDB-Logging aufgezeichnet werden, sind Kennziffern von Garbage-Collector-Läufen, Thread-Anzahlen, belegter Arbeitsspeicher, CPU-Last oder Ausführungsdauern bestimmter Aktionen.

**Livedatenanzeige in der Komponente Cockpit**

Das *Cockpit* ist eine Library, die in allen iWACS®-Modulen verwendet werden kann. Sie ermöglicht es dem Modul, bestimmte sogenannte Kennzahlen (die vom Modul programmatisch an der Cockpit-Komponente angemeldet werden) mithilfe von verschiedenen Diagrammen zu visualisieren und ist im Wesentlichen zur Anzeige von Livedaten und aktuellen Zuständen konzipiert.

Im Client eines iWACS®-Moduls, in dem die Cockpit-Library verwendet wird, steht dem Benutzer der Cockpit-Dialog zur Verfügung. Hier wird eine Liste aller registrierten Kennzahlen angezeigt. Für jede Kennzahl können mehrere Widgets angelegt werden, um die (aktuellen) Werte der Kennzahlen zu visualisieren; beispielsweise Torten-, Balken- oder Liniendiagramme, aber auch Tabellen oder eine simple textuelle Ausgabe des aktuellen Werts.

Typische Daten, die im Cockpit angezeigt werden, sind Durchsätze von Fördererstrecken, Fehllesungen von Scannern, Füllstände von Puffern oder Lagern, offene oder abgearbeitete Aufträge oder Wareneingänge seit Tagesbeginn.

**Umgebungsdatenüberwachung im Modul .MON**

Das Modul iWACS®.MON kann bestimmte Kennzahlen einer iWACS®-Installation überwachen und aufzeichnen. Dazu werden diese Werte durch konfigurierbare Prüfungen, beispielsweise auf Schwellwertüberschreitungen geprüft. Wenn eine konfigurierte Prüfung einmal oder mehrmals fehlschlägt, kann .MON beispielsweise zugeordnete E-Mail-Benachrichtigungen versenden.

.MON kann verschiedene Arten von Daten aufzeichnen – beispielsweise Daten, die er über eine vorgegebene Abfrage auf einer Moduldatenbank auslesen kann, oder Werte, die die Module in ihrem jeweiligen Cockpit angemeldet haben. Vornehmlich wird .MON jedoch eingesetzt, um eher hardwarebezogene Kennzahlen zu überwachen, z. B. die CPU-Auslastung von Rechnern, Füllgrade von Datenbanken und Festplatten oder Netzwerklatenzen.

**Reporterstellung im Modul .MIS**

Das Modul iWACS®.MIS dient dazu, in regelmäßigen Intervallen individuelle Berichte über den Zustand der Anlage zu erstellen und zu versenden. Ein Bericht wird dabei immer auf Basis einer sogenannten *Berichtsvorlage* generiert, die der Benutzer des .MIS mit dem separaten Tool *.MIS Report Designer* erstellen kann; hierbei legt er fest, welche Daten und Auswertungen der Bericht in welchem Format (Tabelle, Diagramm usw.) enthalten soll. In einer Berichtsvorlage können verschiedene Daten über grafisch zusammenstellbare oder direkt in JavaScript programmierbare Berechnungen miteinander verknüpft werden.

.MIS kann bei der Generierung eines Berichts auf verschiedene Datenquellen zugreifen; meist verwendet es Daten aus den Archivdatenbanken der verschiedenen iWACS®-Module.

## ANFORDERUNGEN

Um die Erfassung der Anforderungen an eine zentrale Betriebsdatenverwaltung durchführen zu können, müssen zunächst die Ziele des Systems definiert, die Abgrenzung der Komponente zu anderen Aufgaben und Systemen vorgenommen und die Anforderungsquellen für das System ermittelt werden (Rupp 2014, S. 74). Diese Schritte werden im Folgenden beschrieben.

**Ziele und Abgrenzung der Betriebsdatenverwaltung**

In Absprache mit den Entwicklungsleitern von .SAV und iWACS® insgesamt wurden Ziele für die zu realisierende Betriebsdatenverwaltung festgelegt.

Zunächst soll die Betriebsdatenverwaltung, wie bereits beschrieben, dazu dienen, Betriebsdaten von den verschiedenen iWACS®-Modulen zum .SAV zu übermitteln. Gleichzeitig gibt es jedoch auch andere Module, die einen Nutzen aus den gesammelten Daten ziehen können, wie beispielsweise .MIS für Berichte. Allgemein formuliert, soll die Betriebsdatenverwaltung also allen angebundenen Modulen die Möglichkeit bieten, auf die Werte aller Module zuzugreifen.

Neben diesem zentralen Anwendungsfall sollen im Zuge der Realisierung der vereinheitlichten Betriebsdatenverwaltung auch die verschiedenen Varianten und z. T. doppelten Implementierungen zur Speicherung und Verarbeitung von Betriebsdaten in den einzelnen Modulen nach Möglichkeit abgelöst werden. Dies betrifft

- die mittelfristige Speicherung der Daten,
- die korrekte Behandlung von Datenformaten und Einheiten sowie der Semantik der Daten,
- die Überwachung bestimmter Datenreihen z. B. auf Schwellwertüberschreitungen, und
- die Berechnung von aggregierten Werten und Statistiken.

Als weiteres Ziel soll die Betriebsdatenverwaltung größere Auswertungen über alle gesammelten Daten eines längeren Zeitraums ermöglichen, um dadurch systemübergreifende Zusammenhänge besser erkennen zu können. Solche Auswertungen werden voraussichtlich aber nicht am Anlagenstandort, sondern extern durchgeführt. Auch für die Verwendung in .MIS können gesammelte Datenwerte über lange Zeiträume nützlich sein.

Weitere Ziele beschäftigen sich mehr mit den nichtfunktionalen Eigenschaften der Betriebsdatenverwaltung:

- Skalierbarkeit bzgl. der Datenmengen
- Ausfallsicherheit (Vermeidung eines Single Point of Failure)
- gute Performance

- Flexibilität in der Struktur (wichtig bei verschiedenartigen Konfigurationen von Modulen oder Ressourcen in unterschiedlichen Anlagen)
- Keine Beeinträchtigungen des regulären Betriebs der Module
- Benutzbarkeit

Diese Ziele stecken grob die Funktionalitäten ab, die die Betriebsdatenverwaltung leisten muss. Funktionen, die über den hier beschriebenen Bereich hinausgehen, sollen (zumindest vorerst) nicht integriert werden und grenzen so die zu entwickelnde Betriebsdatenverwaltung ab. Dies betrifft beispielsweise

- Komponenten zur Visualisierung der Daten
- Unterstützung der augenblicklichen Kommunikation von Livedaten (z. B. zu aktuell laufenden Vorgängen oder Aktionen)
- Übertragung von Daten, die keine Beschreibung von Vorgängen in der Anlage darstellen, sondern z. B. diese steuern
- Übertragung von nichtnumerischen Daten wie Texte

### Ermittlung der Anforderungen

Um Anforderungen an die Betriebsdatenverwaltung erarbeiten zu können, müssen zunächst alle relevanten Anforderungsquellen ermittelt werden. Dies ist eine zentrale Aufgabe innerhalb des Requirements Engineering: Wenn wichtige Anforderungsquellen übersehen werden, können deren Anforderungen nicht dokumentiert werden, was dazu führen kann, dass das System später wichtige Funktionalitäten nicht abdeckt. Mögliche Anforderungsquellen sind dabei Dokumente (Handbücher zu Altsystemen, Prozessbeschreibungen), Systeme (Vorgängersysteme, Nachbarsysteme) und Stakeholder (Personen oder Organisationen, die Einfluss auf die Anforderungen haben) (Rupp 2014, S. 77).

Die zu untersuchenden Dokumente umfassten insbesondere Konzepte für Entwicklung und Einsatz des neuen .SAV sowie Dokumentationen zu bestehenden Modulen und Tools.

Bedingt durch das Prinzip der Betriebsdatenverwaltung mit zahlreichen angebundenen Komponenten existiert auch eine Reihe von Systemen, die Anforderungen an die Betriebsdatenverwaltung stellen – sowohl Vorgängersysteme, die in der Ist-Analyse beschrieben wurden, als auch benachbarte Systeme, die an die Betriebsdatenverwaltung angebunden werden sollen.

Bei der Realisierung der Betriebsdatenverwaltung sind zudem auch die Stakeholder mit einzubeziehen, die Anforderungen an diese stellen können. Die Ermittlung der Stakeholderanforderungen ist wichtig, da bei der Betriebsdatenverwaltung wie bei jedem System mehrere Personen bzw. Gruppen betroffen sind, die ganz unterschiedliche Sichten auf das System haben und unterschiedliche – z. T. sogar gegenläufige – Zielsetzungen verfolgen und Ansprüche stellen können (Rupp 2014, S. 79). Die Stakeholder der Betriebsdatenverwaltung beinhalten u. a. den .SAV-Produktmanager, die Entwickler der einzelnen iWACS®-Module, die Realisierungs- und die Serviceabteilung der TSS sowie den Endkunden.

Um alle Anforderungen an die Betriebsdatenverwaltung zusammenstellen zu können, mussten daher die betreffenden Personen befragt und die beschriebenen Dokumente und Systeme auf Anforderungen untersucht werden. Dabei entstanden die sogenannten *Stakeholderanforderungen*, also Funktionen, die konkret von einem Stakeholder benötigt werden oder deren Notwendigkeit aus einem Dokument bzw. einem bestehenden System ersichtlich ist.

Aus den Stakeholderanforderungen wurden anschließend *Produktanforderungen* abgeleitet, die die Stakeholderanforderungen konsolidieren und vor allem auch spezifizieren, wie diese durch das System umgesetzt werden sollen.

## MODELLENTWURF ZUR ABBILDUNG VON BETRIEBSDATEN

Dieser Abschnitt befasst sich mit den Ergebnissen der Anforderungsanalyse hinsichtlich der Entitäten, Beziehungen und Abläufe, die die Betriebsdatenverwaltung aus Sicht der angebundenen Module abbilden können muss. Hierzu wird Schritt für Schritt ein Modell vorgestellt, das die Erkenntnisse aus den entsprechenden Anforderungen zusammenfasst. Insbesondere werden hierbei auch einige neue Begriffe eingeführt und unscharf verwendete Begriffe genauer definiert.

### Datenelemente

Der Begriff *Betriebsdaten* wurde bereits definiert als „sämtliche Daten, die vergangene Ereignisse, Vorgänge oder Zustände in einer Logistikanlage quantitativ beschreiben“. Betriebsdaten sind demzufolge im Wesentlichen eine Sammlung von numerischen Werten. Ein einzelnes („Betriebs-“) Datum, d. h. ein einzelner Zahlenwert, den die Module über die Betriebsdatenverwaltung veröffentlichen und abfragen können sollen, wird im Folgenden als *Datenelement* bezeichnet; synonym wird manchmal auch einfacher *Wert* genutzt.

### Metriken

Die Datenelemente, die die Betriebsdatenverwaltung sammeln soll, werden in Zeitreihen erfasst; d. h. eine bestimmte Größe wird im Zeitverlauf in regelmäßigen oder unregelmäßigen Abständen bestimmt und abgespeichert. Solche Größen werden im Folgenden *Metriken* genannt; ein Datenelement ist also immer einer Metrik fest zugeordnet. Die Metrik eines Datenelements stellt somit eine Beschreibung der inhaltlichen Aussage dieses Datenelements dar. Jedes Datenelement kann über die zugeordnete Metrik und seine Zeitangabe eindeutig identifiziert werden.

Beispiele für Metriken sind:

- Kommissionierte Menge an einem Arbeitsplatz bei der Bearbeitung von TEs vom Typ X durch einen bestimmten Lagermitarbeiter

- Anzahl ausgelagerter TEs innerhalb von 24 Stunden in der gesamten Anlage
- Durchschnittliche Temperatur eines Motors nach Tagen aufgeschlüsselt
- Kumulierter Zeitaufwand für Garbage Collection seit dem Start eines bestimmten Moduls
- Anzahl Prozessoren eines Rechners, die in der letzten halben Stunde länger als 10 Minuten eine Auslastung größer 90% aufwiesen

Von entscheidender Bedeutung für die Strukturierung der gesammelten Daten innerhalb der Betriebsdatenverwaltung ist die Art und Weise, wie Metriken definiert und klassifiziert werden; dies beeinflusst auch, wie flexibel die gesammelten Daten später kombiniert, aggregiert und analysiert werden können.

Denkbar wäre, die Metriken einer Anlage in einer Art Baumstruktur von Namespaces einzuordnen. Da zwischen den einzelnen Unterscheidungskriterien allerdings oftmals kein hierarchischer Zusammenhang besteht, ist es sinnvoller, hier eine nicht-hierarchische Gliederung einzusetzen. Dazu muss auf jeder Metrik zur eindeutigen Identifikation eine Liste von *Tags* gespeichert werden. Jeder Tag besteht aus einem Key-Value-Paar; der Key ist dabei ein *Merkmal* wie „Modul“, „Förderer“, „SPS“ oder „Lager“, anhand dessen die Metrik klassifiziert wird; ein Tag *definiert* sein Merkmal auf dieser Metrik. Ein Merkmal kann auf einer Metrik natürlich nicht mehrfach definiert werden. Ähnlich beschreibt auch DIN 1313 ein Merkmal als „eine in objektiver Weise präzisierte Eigenschaft, durch die Objekten, die Träger für das Merkmal sind, jeweils ein Merkmalswert als Kennzeichen der Erscheinungsform zugeordnet wird. (...) Ein Objekt kann Merkmalswerte unterschiedlicher Merkmale tragen, aber von jedem Merkmal kommt ihm nur ein Merkmalswert zu.“ (DIN 1998, S. 15) Der Value eines Tags stellt entsprechend die *Ausprägung* bzw. den Wert des Merkmals für die jeweilige Metrik dar.

Die Tags dürfen sich auch fachlich teilweise überschneiden und damit Redundanzen enthalten, wenn dies z. B. für Aggregationen sinnvoll ist; wichtig ist nur, dass sie in ihrer Gesamtheit je Metrik eindeutig sind. Der Begriff „Tag“ ist hierbei aus dem Kontext der bekannten Zeitreihendatenbank *InfluxDB* entliehen, welche die einzelnen in ihr gespeicherten Metriken (dort als *series* bezeichnet) auf ganz ähnliche Weise unterscheidet (vgl. InfluxData 2018).

**Merkmal „Größe“**

Die *Größe* ist ein spezielles Merkmal, das auf jeder Metrik definiert sein muss.

DIN 1313 definiert die Größe als ein spezielles „Merkmal, für das zu je zwei Merkmalswerten ein Verhältnis gebildet werden kann, das eine reelle Zahl ist.“ (DIN 1998) In diesem Sinne stellt eine Metrik im Kontext der Betriebsdatenverwaltung also einen Schlüssel für eine bestimmte Größe an einem bestimmten Objekt bzw. Merkmalsträger dar, deren Wert sich im Zeitverlauf verändert. Derjenige konkrete Merkmalswert, den die Metrik „messen“ soll, wird durch den Tag mit dem Merkmal Größe festgelegt; das Objekt, auf das sich die Metrik bezieht, wird durch die übrigen Tags definiert.

Beispiele für Ausprägungen des Merkmals „Größe“ auf einer Metrik könnten z. B. sein:

- Elektrische Leistungsaufnahme, Ausführungsdauer
- Kontext Fahrzeugantriebe: Betriebsstundenzähler, Kilometerzähler, aktuelle Position, aktuelle Geschwindigkeit, aktuelles Drehmoment
- Kontext Systemüberwachung: CPU-Last, Netzwerklatenzzeit, Übertragungsrate, Disk-IO, DB-Füllgrad, Prozess aktiv
- Kontext Materialfluss: Kreuzungsdurchsatz, Anzahl Fehllesungen

Jeder Größe ist fest eine Dimension (d. h. beispielsweise Länge, Geschwindigkeit, Anzahl, Gewicht, ...) zugeordnet. Dadurch kann die Betriebsdatenverwaltung auch bei Berechnungen informiert reagieren und z. B. die Dimensionen (und damit auch Einheiten) von Ergebnissen automatisch bestimmen. (DIN 1998, S. 6) Die Betriebsdatenverwaltung muss dazu die sieben SI-Basisdimensionen sowie die Dimensionen *Zahl* und *Datenmenge* kennen. Alle abgeleiteten Dimensionen gehen dann auf ein Produkt von Potenzen der Basisdimensionen zurück. (DIN 1998, S. 8)

Die Größe einer Metrik ist auch wichtig, um mögliche Aggregationen bestimmen zu können. Multiplikationen und Divisionen zwischen verschiedenen Größen (oder selbst Dimensionen) sind prinzipiell immer denkbar (wenn auch nicht immer physikalisch sinnvoll). Addition oder Subtraktion zwischen Werten verschiedener Dimensionen sind zwar nicht möglich; ob diese Operationen zwischen verschiedenen *Größen* sinnvoll sind, kann jedoch oft nicht eindeutig definiert werden. Ein (physikalisches) Beispiel hierfür sind die Größen „Arbeit“ und „Drehmoment“. (Wallot 1953, S. 140) Beide haben die Dimension $M \cdot L^2 \cdot T^{-2}$ bzw. „Kraft mal Länge“, jedoch leitet sich dies im Fall der Arbeit von einer Kraft ab, die über diese Länge hinweg wirkt, während beim Drehmoment die Länge eines Hebels, an dem die Kraft wirkt, gemeint ist; um das Drehmoment in eine Energie zu überführen, ist zusätzlich noch die Angabe des Drehwinkels des Hebels notwendig. Ebenso macht es im Kontext einer Logistikanlage wenig Sinn, z. B. die Anzahl der aktiven SPS-Störungen, die Anzahl TEs in einem Lager und die Anzahl Scanner-Fehllesungen innerhalb der letzten Stunde zu addieren, auch wenn es sich jeweils um Größen der Dimension „Zahl“ handelt. Umgekehrt kann es jedoch durchaus sinnvoll sein, etwa die Anzahl offener Auslageraufträge von der Anzahl TEs in einem Lager zu subtrahieren. Aus diesem Grund sind Addition und Subtraktion zwischen Metriken verschiedener Größen, aber gleicher Dimension, in der Betriebsdatenverwaltung grundsätzlich zugelassen.

### Erfassungen

Aus den Anforderungen an die Betriebsdatenverwaltung ergibt sich, dass drei grundlegende Arten der Datenentstehung berücksichtigt werden müssen: aktiv, passiv und berechnend.

Die Art, wie die Datenelemente einer Metrik entstehen, ist nicht zwingend für alle Datenelemente der Metrik gleich; daraus folgt, dass diese nicht an der Metrik, sondern an einer weiteren Entität festgemacht werden muss, die die Art der Datengewinnung für eine bestimmte Metrik in einem bestimmten Zeitraum beschreibt. Diese Entität wird im Folgenden als *Erfassung* bezeichnet.

Diese Aufteilung ist insbesondere deswegen notwendig, weil auch denkbar ist, dass eine Erfassung mehrere Metriken gleichzeitig mit Datenelementen versorgen kann. Dies betrifft z. B. Erfassungen, die mehrere Werte in einer einzelnen Datenbanktransaktion abfragen, oder Berechnungen, bei denen mehrere Ergebnisse entstehen.

Bei einer **passiven Erfassung** werden die Daten aus einer Logik innerhalb eines Moduls heraus gemeldet; dies ist z. B. bei der Aufzeichnung von Ereignissen der Fall.

**Aktive Erfassungen** werden verwendet, wenn die Betriebsdatenverwaltung nach Anweisung eines Moduls selbständig regelmäßig Werte für eine Metrik ermittelt; dies ist vor allem bei der Abtastung von Zuständen nützlich und wird in den bestehenden Lösungen (Cockpit, RRDB) bereits vielfach angeboten. Das Modul muss hierzu der Betriebsdatenverwaltung die Abfragehäufigkeit und die Logik für die Abfrage mitteilen; anschließend ist die Betriebsdatenverwaltung dafür verantwortlich, die Abfragelogik entsprechend den geforderten Intervallen aufzurufen.

Prinzipiell könnte ein solcher Mechanismus zur regelmäßigen Abfrage auch durch die Module selbst implementiert werden; jedoch würde dies wieder Aufwände und Duplizierungen in den einzelnen Modulen verursachen.

**Berechnende Erfassungen**, oder kurz Berechnungen, führen Transformationen und Aggregationen der Daten durch. Diese Erfassungen generieren also ihre Datenelemente vollständig innerhalb der Betriebsdatenverwaltung, indem sie sie aus anderen, bereits bekannten Datenelementen berechnen.

### Löschkonfigurationen

Die gesammelten Beobachtungen werden in der Praxis nicht für immer aufbewahrt werden können; stattdessen ist es erforderlich, aus Kapazitätsgründen alte Daten nach einiger Zeit zu löschen oder zusammenzufassen. Dies soll die Betriebsdatenverwaltung über sogenannte *Löschkonfigurationen* ermöglichen.

Hierzu muss die Betriebsdatenverwaltung alle Metriken regelmäßig überprüfen. Wenn zu einer Metrik in einem bestimmten Zeitraum mehr Beobachtungen gespeichert sind als nach der zugeordneten Löschkonfiguration gespeichert sein sollten, werden diese mit einer in der Löschkonfiguration angegebenen Funktion aggregiert, die Ergebnisse in einer zugeordneten Metrik als Beobachtungen abgelegt und die alten Beobachtungen werden gelöscht.

### Abfragen und Abonnements

Um die gesammelten Daten nutzen zu können, müssen die Module Möglichkeiten haben, diese abzufragen. Hierbei werden grundsätzlich zwei Varianten unterschieden: Abfragen von Datenelementen eines definierten Zeitraums (der Vergangenheit) und Abonnements von Metriken.

Bei der Abfrage eines Datenelements aus einem definierten Zeitraum gibt das Modul die Metrik an, aus der es Datenelemente benötigt, sowie den gewünschten Zeitraum, die gewünschte Einheit und die gewünschte Auflösung; die Betriebsdatenverwaltung liefert dann die benötigten Daten zurück.

Ein Modul kann außerdem eine Metrik abonnieren, um bei der Entstehung neuer Datenelemente dieser Metrik benachrichtigt zu werden. Wenn ein Modul beendet wird, verfallen seine Abonnements.

## KONZEPTAUSWAHL ZUR GROBSTRUKTUR DER IMPLEMENTIERUNG

Dieser Abschnitt stellt nun zunächst eine Reihe von Varianten zum grundlegenden technischen Aufbau eines Betriebsdatenverwaltungssystems vor; anschließend werden die Konzepte gegeneinander abgewogen, um zuletzt eine Empfehlung auszusprechen.

Grundsätzlich fallen im Rahmen der Betriebsdatenverwaltung Zeitreihendaten und Metadaten an. Die Zeitreihendaten entsprechen dabei den Datenelementen, die die Betriebsdatenverwaltung in großer Anzahl speichern und verwalten muss, während alle übrigen Informationen (Metriken, Merkmale, Merkmalsausprägungen, Erfassungen usw.) in die Kategorie der Metadaten fallen.

In jeder Konstellation ist es notwendig, dass die Schnittstelle zwischen den einzelnen Modulen und der Betriebsdatenverwaltung über eine Programmlibrary realisiert ist, sodass aus Sicht der Module zur Interaktion ausschließlich direkte API-Aufrufe notwendig sind. So kann sichergestellt werden, dass die Meldung von Beobachtungen durch die Module nicht blockiert und dass kein redundanter Code in den Modulen nötig ist. Alle iWACS®-Module sind in Java implementiert, sodass hier für alle Module dieselbe Bibliothek eingesetzt werden kann. Wenn im Folgenden von Aktionen der Module gesprochen wird, so sind damit auch Aktionen gemeint, die durch die eingebundene Betriebsdaten-Library in den Modulen durchgeführt werden.

### Metadatenverwaltung

Zunächst sollen zwei Möglichkeiten beschrieben werden, um die Metadaten innerhalb einer Installation der Betriebsdatenverwaltung zu speichern. Die Metadaten sind zwar erheblich weniger umfangreich als die Zeitreihendaten, jedoch

ist ihre Verwendung und ganz allgemein ihr Lebenszyklus deutlich komplexer: Sie müssen über alle Module hinweg konsistent bekannt sein, und viele Metadatensätze können nach dem Anlegen auch bearbeitet und evtl. wieder gelöscht werden.

Die einfachste und naheliegendste Option ist es, die Metadaten je Installation in einer einzelnen, zentralen Datenbank zu speichern, die klassischerweise relational aufgebaut und unabhängig von den Modulen ist. Somit ist sichergestellt, dass nur ein einziger, konsistenter und persistent abgelegter Datenstand existiert. Die Absicherung der Konsistenz und die Verhinderung von parallelen Zugriffen durch mehrere Module auf einen Datensatz erfolgt durch die Concurrency-Bordmittel der Datenbank, die diese im benötigten Umfang mitbringen muss.

Eine zweite Variante wäre, dass jedes Modul einen eigenen Metadatenspeicher hält und die Module (bzw. genauer die Betriebsdaten-Libraries) die Synchronisierung übernehmen. Diese Datenbank könnte auch einfach in die bestehende Datenbank des jeweiligen Moduls integriert werden. Der komplexeste Teil an dieser Lösung ist der notwendige Synchronisationsmechanismus, der sicherstellt, dass die Datenstände der einzelnen Module sich nicht widersprechen. Dieser erfordert ein hohes Maß an Kommunikation zwischen den Modulen und kann insbesondere zu Problemen führen, wenn die Verbindung eines Moduls nach außen abbricht und es sich nicht mit den anderen Modulen abstimmen kann.

In Abbildung 1 werden die beiden Konzepte anschaulich schematisch gegenübergestellt: Links die dezentrale, rechts die zentrale Metadatenspeicherung. Die roten Datenbanksymbole und Pfeile stellen Speicher bzw. Flüsse von Metadaten dar; die Library zur Betriebsdatenverwaltung, die in die Module eingebunden wird, ist hier mit „BDV-Library“ abgekürzt.

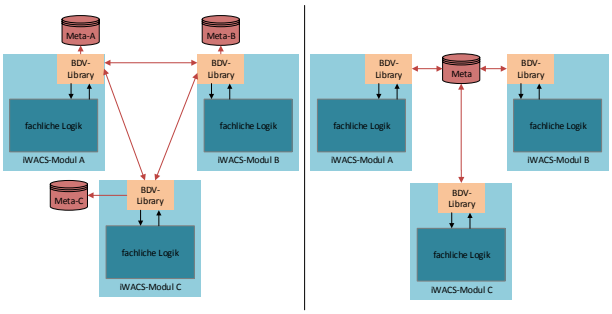


Abbildung 1. Mögliche Strukturen zur Verwaltung der Metadaten

Die dezentrale Organisation hat gegenüber der zentralen Form einige Vorteile:

- Die Module können autarker arbeiten, da jedes eine Kopie der Metadatenbasis zur Verfügung hat.
- Es existiert kein Single Point of Failure.
- Es muss nicht zwingend eine separate Datenbank für die Metadaten eingerichtet werden; stattdessen kann das Schema zur Metadatenverwaltung auch in die – sowieso notwendige – Datenbank des iWACS®-Moduls integriert werden.
- Je nach Ausgestaltung des Synchronisationsmechanismus können Netzwerkzugriffe prinzipiell flexibel so früh oder so spät wie möglich vorgesehen werden (*lazy loading* oder *eager loading*).
- Falls die Module räumlich verteilt installiert sind (z. B. einige Module in einem Serverraum direkt in der Anlage und andere Module in einem entfernten Rechenzentrum), kann die jeweilige Metadaten-DB beim Modul mit installiert werden, was Zugriffszeiten und Durchsätze verbessert.

Diesen Punkten steht aber auch eine Reihe von Nachteilen gegenüber:

- Das nötige Synchronisationsprotokoll zwischen den Modulen ist sehr komplex und würde im Wesentlichen eine Nachimplementierung von bereits existierenden verteilten Datenbankanwendungen darstellen; hingegen sind die Concurrency-Funktionalitäten vieler verbreiteter Datenbanken bewährt und ausgereift.
- Die Struktur des Systems ist komplex und die Abläufe sind schwierig zu verstehen, was die Analyse von Fehlern und Problemen erschwert.
- Bei Abfragen z. B. aller Metriken ist oft viel (zeitaufwändige) Kommunikation mit zahlreichen Partnern notwendig, bevor ein Ergebnis vorliegt.
- Es gibt keine zentrale Stelle, an der bestimmte systemweite Konfigurationen wie Verbindungsdaten einheitlich gepflegt werden könnten.

Aufgrund dieser Nachteile für die Realisierung der Metadatenverwaltung wird die erstgenannte **zentralisierte Variante** empfohlen. Diese Entscheidung beruht auf der Komplexität und den unvermeidbaren Problemen bei der Umsetzung des Synchronisationsprotokolls. Hingegen relativieren sich die Vorteile der dezentralen Option bei genauerer Betrachtung zumindest teilweise wieder:

- Autarkes Arbeiten ist nur so lange möglich, wie keine Kommunikation benötigt wird; diese kann aber den meisten Fällen (je nach Ausgestaltung der Synchronisation) nicht vermieden werden.
- Ausfallsicherheit kann auch bei (aus Benutzersicht monolithischen) relationalen Datenbanken z. B. durch Replizierung erreicht werden.
- Bei räumlicher Verteilung der Module kann das dezentrale Modell zwar die Zugriffszeiten auf die Metadaten-DB verbessern, gleichzeitig dürfte aber die verlangsamte Kommunikation bei der Synchronisierung diesen Vorteil wieder zunichtemachen.

**Zeitreihenverwaltung**

Die Zeitreihendaten sind im Vergleich zu den Metadaten deutlich einfacher strukturiert: Im Wesentlichen handelt es sich um große Mengen an Datenelementen, die nur hinzugefügt, aber nie bearbeitet werden. Stattdessen stellt hier die Datenmenge die eigentliche Herausforderung dar; Schätzungen ergaben, dass bei großen Anlagen und intensiver Sammlung von Daten bis zu 20.000 Datenelemente pro Sekunde im Durchschnitt (!) zu erwarten sind.

Der Anwendungsfall, größere Mengen an Zeitreihendaten zu speichern und auszuwerten, kommt auch außerhalb der Intralogistik relativ häufig vor. Aus diesem Grund existiert bereits eine große Anzahl an speziell dafür konzipierten Zeitreihendatenbanken; Beispiele hierfür sind *InfluxDB*, *OpenTSDB*, *Prometheus* oder *Riak TS*. Diese sind auf die oben beschriebene Verwendung hin optimiert und erlauben insbesondere auch oft bereits eine Vielzahl an Aggregationen über die gespeicherten Daten, sodass diese direkt in der Datenbank ausgeführt werden können; dies dürfte die Ausführungszeiten für solche Operationen erheblich senken. Außerdem verwenden diese Datenbanken oft Datenstrukturen, die dem vorgestellten Modell sehr ähnlich sind, was die Ablage der Daten stark erleichtert. Es empfiehlt sich also, für die Betriebsdatenverwaltung ebenfalls eine solche Datenbank einzusetzen.

Zudem muss die Betriebsdatenverwaltung bei jeder Entstehung eines neuen Datenelements (sowohl bei Beobachtungen als auch bei Derivaten) eventuell andere Module benachrichtigen (z. B. Module, die die Metrik des Datenelements abonniert haben, oder die das Datenelement für die Aktualisierung von aggregierten Werten benötigen).
*1) Zentrale Datenhaltung:* Der naheliegendste Ansatz ist auch in diesem Fall der Einsatz einer einzelnen, zentralen Datenbank, die die Beobachtungen sämtlicher Module aufnimmt. Alle Module (bzw. die Betriebsdatenverwaltung-Libraries) senden ihre Beobachtungen direkt an diese Datenbank und fragen sie direkt dort ab. Notwendige Aggregationen und Berechnungen werden bereits durch die Datenbank durchgeführt, d. h. die Datenmenge, die ein einzelnes Modul von der Datenbank abfragt, hält sich in den meisten Fällen in engen Grenzen. In der Library selbst muss daher in diesem Szenario im Idealfall auch keine Logik für Aggregationen implementiert sein.

Falls die Zeitreihendatenbank dies zulässt, könnte das Schema für die Metadaten auch in diese Datenbank mit integriert werden; somit wäre für das gesamte System nur eine einzige zusätzliche Datenbank notwendig.

Diese Datenbank muss so ausgelegt sein, dass sie die anfallende Datenmenge verarbeiten kann: sowohl das Gesamtvolumen, als auch die Einfügungsrate. Die Datenbank sollte Hardwareressourcen nicht mit einem Modul teilen müssen, um gegenseitige Beeinträchtigungen zu verhindern.

Berechnungen werden durch diejenigen Module angestoßen, die die Werte benötigen (also die die Zielmetriken der Berechnung abonniert haben oder Werte aus ihnen abfragen). Somit ist sichergestellt, dass jedes Modul alle benötigten aggregierten Daten immer ermitteln kann, wenn die Zeitreihendatenbank erreichbar ist. Nachteilig ist jedoch, dass eventuell dieselbe Berechnung von mehreren Modulen einzeln angefordert wird. Manche Zeitreihendatenbanken (z. B. InfluxDB (InfluxData 2018) oder OpenTSDB (OpenTSDB 2016)) unterstützen auch automatisierte, regelmäßige Aggregationen innerhalb der Datenbank, um dieses Problem zu entschärfen.

Wenn ein Modul eine neue Beobachtung meldet, muss es auch die nötigen Benachrichtigungen an alle Module versenden; für diesen Zweck muss es Verbindungen zu allen anderen Modulen unterhalten.

Optional ist es denkbar, vor die zentrale Datenbank noch einen separaten Prozess zu platzieren, der zwischen den Modulen und der Datenbank vermittelt. Dieser könnte Berechnungen übernehmen, die die Datenbank nicht unterstützt und die aufgrund des hohen Berechnungsaufwands nicht durch das anfragende Modul durchgeführt werden sollten. Außerdem könnten Benachrichtigungen der Module sowie Berechnungen zentral über diesen Prozess abgewickelt werden, sodass überhaupt keine Verbindungen mehr zwischen den Modulen untereinander notwendig sind. Andererseits erhöht ein solcher zusätzlicher Prozess wieder die Komplexität des Gesamtsystems und verschlechtert Durchsätze und Latenzen, da alle Datenbankzugriffe über eine weitere Zwischenstation laufen müssen.
*2) Dezentrale Datenhaltung:* Den Gegensatz zur gerade beschriebenen Variante stellt eine vollständig dezentrale Datenhaltung dar. Dies bedeutet, dass für jedes Modul eine eigene Zeitreihendatenbank angelegt wird, in der alle Beobachtungen gespeichert werden, die durch dieses Modul erfasst werden. Diese Datenbank wird allein durch das jeweilige Modul verwaltet; dadurch muss sie auch keine parallelen Zugriffe mehrerer Prozesse verarbeiten können.

Benachrichtigungen an andere Module sendet jedes Modul auch hier selbständig.

Bei Abfragen muss das interessierte Modul zunächst ermitteln, welches Modul die benötigten Daten erfasst hat und sie dann von diesem Modul anfordern. Aggregationen, die über Daten aus mehreren verschiedenen Modulen laufen, müssen dann zumindest teilweise durch das anfragende Modul erledigt werden; d. h. die Daten der einzelnen Module werden zunächst durch das erfassende Modul und seine Datenbank soweit wie möglich aggregiert, und die Ergebnisse werden an das anfragende Modul übertragen, welches dann die verbleibenden Berechnungen durchführt.

Auch hier ist wichtig, dass den Zeitreihendatenbanken entsprechende Hardwareressourcen zur Verfügung stehen, die sich nicht mit denen der Module selbst überschneiden.

Berechnungen müssen auch in dieser Konstellation durch jedes Modul, das eine Zielmetrik einer Berechnung abonniert hat, selbständig angestoßen werden. Eventuell können die einzelnen Module Ergebnisse von Abfragen einige Zeit zwischenspeichern, um mehrfache Berechnungen zu vermeiden.

Alternativ wäre auch denkbar, dass die Module bei Abfragen aus Zeitreihendatenbanken anderer Module direkt auf diese Datenbanken zugreifen. Dies könnte Kommunikationsoverhead einsparen, erfordert dann jedoch, dass jedes Modul pro anderem Modul *zwei* Verbindungen verwalten muss.
*3) Gemischte Datenhaltung:* Die dritte Option, die hier vorgestellt werden soll, stellt eine Kombination aus zentraler und dezentraler Datenspeicherung dar.

Hierbei werden sowohl eine zentrale Zeitreihendatenbank, als auch verteilte Datenbanken pro Modul angelegt. Bei der Entstehung von Beobachtungen werden diese durch das Modul in dessen eigene, private Datenbank und zusätzlich noch in die zentrale Datenbank eingetragen. Dies könnte über zwei Einfügeoperationen auf den beiden Datenbanken gelöst werden, oder auch durch die Datenbanken selbst, wenn diese eine solche Funktion (Spiegelung aller Einfügeoperationen an eine andere Datenbankinstanz) unterstützen.

Abfragen werden in diesem Szenario wie bei der zentralen Datenhaltung auf der zentralen Datenbank ausgeführt. Hierfür ist insbesondere wichtig, dass die Datenbanken immer synchron gehalten werden, um fehlerhafte Ergebnisse zu vermeiden (z. B. wenn eine Berechnung aufgrund einer neuen Quellbeobachtung aufgerufen wird, aber die Quellbeobachtung noch nicht in der zentralen Datenbank abgelegt wurde). Falls Netzwerkprobleme auftreten sollten, können die einzelnen Module immer noch regulär Daten einspeisen und auch die eigenen Daten abfragen.

Auch in dieser Variante ist denkbar, vor die zentrale Datenbank einen separaten Prozess zur Vermittlung zu schalten.
*4) Konzeptempfehlung:* Um die drei vorgestellten Konzepte und ihre Varianten übersichtlich miteinander vergleichen und bewerten zu können, wurden diese zusammen mit ihren Vor- und Nachteilen in einer Entscheidungsmatrix gegenübergestellt.

Auf dieser Basis habe ich entschieden, die Variante „Zentrale Datenhaltung“ mit einem Zentralprozess als beste Option zu empfehlen. Diese Entscheidung basiert vor allem auf folgenden Gründen:

- Die Anzahl notwendiger Systeme und Verbindungen ist vergleichsweise gering, da nur eine Zeitreihendatenbank und ein einzelner zentraler Prozess benötigt wird. Dies reduziert die Komplexität der Lösung deutlich im Vergleich zu anderen Varianten.
- Aggregationen können zentral durchgeführt und koordiniert werden. Dadurch ist eine Belastung der Module durch aufwändige Berechnungen und große zu übertragende Datenmengen ausgeschlossen. Zudem ist keine Abstimmung über die Durchführung von Aggregationen notwendig und Aggregationen müssen nicht auf mehrere Datenbanken aufgesplittet werden, was ebenfalls die Komplexität des Systems ganz erheblich vermindert.
- Für Datenabfragen bzw. Benachrichtigungen aufgrund von Abonnements ist zwar eine Verbindung zum Zentralprozess notwendig; zumindest die Benachrichtigungen, die nicht auf berechnenden Erfassungen basieren, könnten aber über Direktverbindungen auch bei einem Ausfall des Zentralprozesses übermittelt werden. Das Melden neuer Beobachtungen muss aufgrund hoher Performanceanforderungen ohnehin gepuffert werden und ist daher (zumindest übergangsweise) auch ohne Verbindung zum Zentralprozess möglich.
- Es werden keine Daten mehrfach gehalten, was den Speicherplatzbedarf reduziert.

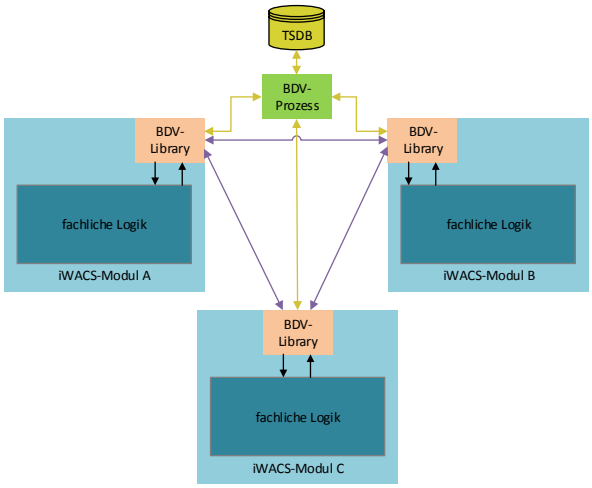


Abbildung 2. Empfohlene Struktur zur Verwaltung der Zeitreihendaten

Die empfohlene Struktur ist in Abbildung 2 grafisch dargestellt. Die gelben Pfeile stellen Flüsse von Zeitreihendaten dar; die violetten hingegen stehen für die (optionalen) direkten Benachrichtigungen zwischen den Modulen. Die Metadatenverwaltung ist hier zur besseren Übersichtlichkeit nicht mit abgebildet.

## FAZIT

Im Rahmen dieser Arbeit wurden die ersten Schritte hin zu einer einheitlichen Erfassung, Verwaltung und Kommunikation von Betriebsdaten in iWACS®-Intralogistiksystemen unternommen.

Dies beinhaltete zunächst die Analyse des Istzustandes in diesem Bereich innerhalb der iWACS®-Plattform. In einem zweiten Schritt erfolgte dann die Anforderungsanalyse, in der aus den Zieldefinitionen, aus Interviews mit Stakeholdern und aus der Untersuchung von Dokumenten und älteren

Systemen Anforderungen an die Betriebsdatenverwaltung gewonnen wurden.

Anschließend wurde ein Datenmodell entwickelt und vorgestellt, mit dem Betriebsdaten einer Intralogistikanlage so abgebildet werden können, dass die ermittelten Anforderungen möglichst gut abgedeckt werden.

Zuletzt erfolgte eine Untersuchung möglicher Konzepte zur Strukturierung der Betriebsdatenverwaltung, von denen schließlich für eines eine Empfehlung ausgesprochen wurde.

Diese Ergebnisse dienen als Grundlage für das weitere Vorgehen zur Entwicklung der Betriebsdatenverwaltung. Als nächstes wird hierzu ein erster Prototyp entworfen und implementiert, mit dem durch Tests dann untersucht werden kann, welche Konzepte und Technologien für die Umsetzung der Betriebsdatenverwaltung eingesetzt werden sollten; dies betrifft z. B. die verwendete Zeitreihendatenbank und die Technologien zur Kommunikation der einzelnen Komponenten des Systems untereinander.

## REFERENCES


DIN (Dez. 1998). *DIN 1313 – Größen*. Norm. Deutsches Institut für Normung e. V.

InfluxData (Nov. 2018). *InfluxDB 1.7 Documentation*. InfluxData. URL: https://docs.influxdata.com/influxdb/v1.7/ (besucht am 19. 11. 2018).

OpenTSDB (2016). *OpenTSDB 2.3 documentation: Rollup And Pre-Aggregates*. URL: http://opentsdb.net/docs/build/html/user_guide/rollups.html (besucht am 21. 12. 2018).

Rupp, Chris (2014). *Requirements-Engineering und -management*. 6. Aufl. München: Carl Hanser Verlag.

Wallot, Julius (1953). *Grössengleichungen, Einheiten und Dimensionen*. Leipzig: Johann Ambrosius Barth Verlag.