Fakultät für Informatik
Institut für Verteilte Systeme
Lehrstuhl Echtzeitsysteme und Kommunikation

Bachelorarbeit

Entwicklung und Umsetzung einer standardkonformen Kommunikation für Batteriespeicher

Autor:

Markus Wolf

26. März 2015

Prüfer:	Prof. Dr. Edgar Nett
Betreuer:	Dipl.-Ing.-Inf. Timo Lindhorst
Praktikumsbetreuer:	Dipl.-Inform. Kathleen Hänsch
	Dr. André Naumann

Wolf, Markus:
Entwicklung und Umsetzung einer standardkonformen Kommunikation für Batteriespeicher
Bachelorarbeit, Otto-von-Guericke-Universität Magdeburg, 2015.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DNP3	Distributed Network Protocol 3
EEX	European Energy Exchange
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IKT	Informations- und Kommunikationstechnologien
IP	Internet Protocol
MMS	Manufacturing Messaging Specification
PV	Photovoltaik
REST	Representational State Transfer
SCADA	Supervisory Control and Data Acquisition
SCL	Substation Configuration Language
SoC	State of Charge
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
VMD	Virtual Manufacturing Device
WAN	Wide Area Network
XML	Extensible Markup Language

1 Einleitung

Zur Zeit befindet sich das elektrische Netz, nicht nur in Deutschland, sondern auch in vielen anderen Regionen der Welt, im Wandel. Die Ursache hierfür liegt in dem Bestreben, insbesondere in Deutschland im Zuge des Atomausstiegs und der Energiewende, die Energieversorgung in immer größeren Teilen aus regenerativen Energiequellen zu bestreiten. Aufgrund dieser, sowohl politisch geförderten, als auch in der Öffentlichkeit gewünschten, Bestrebung ist eine Zunahme an dezentralen, volatilen Energieerzeugern zu verzeichnen [Nau12]. Jedoch ist bis heute aus Kostengründen keine flächendeckende sowie einheitliche Kommunikation von dezentralen Energieeinspeisern umgesetzt. Das Resultat ist die unkoordinierte Regelung der unteren Netzebenen. Mittelfristig ist also der Wechsel vom passiven zum aktiven Verteilnetz, dem sogenannten Smart Grid, abzusehen [Fab10].

Insbesondere die Erzeugung elektrischer Energie, durch Windkraftanlagen sowie Photovoltaikanlagen, lässt sich nur schwer prognostizieren, geschweige denn regeln. Aufgrund dieser Volalilität der Erzeuger kann keine garantierte Stromversorgung gewährleistet werden, zumal die Netzauslastung im Laufe eines Tages stark schwankt. Um diesem Problem entgegenzuwirken, können Energiespeicher eingesetzt werden, indem sie bei einer Erzeugung von elektrischer Energie, die über dem aktuellen Bedarf liegt (z. B. wenn in Deutschland nachts die Windgeschwindigkeit groß ist, der Verbrauch an Energie jedoch relativ gering [Adr10]), diese Energie speichern und sie dann wieder an das Netz abgeben, wenn es zu Engpässen bei der Stromversorgung kommt. Daneben sind weitere Einsatzszenarien für Energiespeicher denkbar. Wie z. B. die Sicherstellung einer gewissen Versorgungsqualität oder Spannungshaltung, der Betrieb in Micro Grids, oder die Glättung von Spitzenlasten bzw. allgemein dem Lastmanagement [Nau12].

In Deutschland, bedingt durch geologische Gegebenheiten, mangelt es an Pumpspeicherkraftwerken bzw. sind die Möglichkeiten weitestgehend ausgeschöpft. Des Weiteren befindet sich die Technologie ‘Power to Gas’ noch in der Entwicklung und weist ähnlich dem Speichern von elektrischer Energie in Druckluft bis heute noch einen zu geringen Wirkungsgrad auf. So wächst die Nachfrage an Batteriespeichern, welche flexibel einsetzbar sind. Interessant ist insbesondere der Einsatz in Micro Grids.

1.1 Motivation

Infolge der Dezentralisierung und der Zunahme an Systemkomponenten im elektrischen Netz müssen eine Vielzahl von Akteuren koordiniert werden, um dies, sowie einen stabilen Netzbetrieb, zu gewährleisten ist ein umfassender Informationsaustausch unerlässlich. Aktuell verwendete Kommunikationsprotokolle werden dieser Aufgabe oft nicht mehr gerecht. Außerdem müssen, aufgrund der Unterschiede der proprietären Protokolle, Gateways eingerichtet und betrieben werden; dies ist zeitaufwändig, kostet Geld und erhöht die Komplexität des Kommunikationssystems. Aus diesem Grund wurde 1995 damit begonnen, die Normenreihe IEC 61850 zu entwickeln, um die Interoperabilität der Systemkomponenten zu erhöhen. Diese ist dank ihres Konzeptes weltweit anerkannt und dank abstrakter Modellbeschreibung ist ebenfalls ihre Anwendung in der Zukunft gesichert. Jene Normenreihe bzw. jener Standard war zunächst

nur für Stationsautomatisierung vorgesehen, er hat sich jedoch weiter durchgesetzt, sodass nun auch der Datenaustausch von anderen elektrischen Komponenten im Netz beschrieben wird [Nau12].

Zur Zeit ist jedoch die Kommunikation mit Batteriespeichern noch nicht Stand der Technik, soll aber in Zukunft umgesetzt werden. Gerade aufgrund des in den letzten Jahren stark angestiegenden Ausbaus an (privaten) Photovoltaikanlagen (PV-Anlagen) ist das Konzept eines Batteriespeichers interessant, da PV-Anlagen in ihrer Erzeugung von elektrischem Strom stark volatil sind. Bei einer hohen Erzeugung an elektrischem Strom kann dieser gespeichert werden und ist zu einer Zeit, an der zu wenig Strom erzeugt wird, wieder abrufbar. Ein solches beispielhaftes Szenario ist jedoch nur durch einen entsprechenden Datenaustausch zwischen Batteriesystem, PV-Anlage und Kontrollsystem umsetztbar. Insbesondere jener Aufbau soll in der vorliegenden Arbeit, mit besonderem Hinblick auf die Kommunikation mittels IEC 61850, betrachtet werden.

1.2 Aufgabenstellung

Die vorliegende Arbeit soll sich mit der Fragestellung nach der Tauglichkeit des IEC 61850 Kommunikationsstandards für Batteriespeicher auseinandersetzten. Untersucht wird, inwieweit das Informationsmodell sowie definierte Schnittstellen jenes Standards ausreichend sind, um eine Kommunikation zu gewährleisten, die eine vollumfängliche Integration eines Batteriespeichers in das elektrische Netz bzw. in das Smart Grid ermöglicht. 'Vollumfängliche Integration' bedeutet in diesem Kontext, dass die durch die Kommunikation bereitgestellten Informationen hinreichend sind, um eine Batterie im Smart Grid mit allen gewünschten Funktionen steuern bzw. betreiben zu können. Hierzu werden exemplarisch verschiedene Anwendungsfälle betrachtet sowie die Kommunikation zwischen Batteriesystem, PV-Anlage und Leitstelle prototypisch umgesetzt. Des Weiteren soll die im IEC 61850 Standard definierte Beschreibungssprache SCL (Substation Configuration Language) zur Instanziierung der konkreten Datenobjekte Verwendung finden.

Das zu erarbeitende Informationsmodell für den Batteriespeicher soll in einer objektorientierten Programmiersprache umgesetzt werden. Darüber hinaus soll auch das Auslesen der SCL Dateien sowie die darauf aufbauende Instanziierung der Datenobjekte in dieser Sprache erfolgen. An die Programmierung anschließen soll sich ein Betrieb im simulierten System bestehend aus Batterie, PV-Anlage und Leitstelle. Eine Evaluierung der Umsetzung soll abschließend die eingangs beschriebene Fragestellung beantworten.

1.3 Vorgehensweise und Aufbau der Arbeit

In dieser Arbeit werden die für eine Kommunikation mit Batteriespeichern notwendigen Strukturen analysiert, entwickelt und umgesetzt. Zunächst widmet sich das Kapitel 2 den für diese Arbeit wichtigen Grundlagen. Eingebettet in den Kontext des Netzausbaus hin zu einem intelligenten Stromnetz werden im Kapitel 3 verschiedene Geschäftsanwendungsfälle dargestellt. Daran schließt sich die Identifizierung, der für die

Kommunikation relevanten Informationen und Funktionen an. Über die Erarbeitung von Systemanwendungsfällen werden die Geschäftsanwendungsfälle spezifiziert sowie die Verbindungen zwischen den zu kommunizierenden Daten, den Anwendungsfällen und den Anforderungen an die Kommunikation aufgezeigt.

An die Anforderungsanalyse, in der die Anforderungen an die Kommunikation erarbeiten werden, schließt sich das Konzeptkapitel an. In diesem Abschnitt der Arbeit werden verschiedene Lösungsansätze diskutiert und bewertet. Anhand der Anforderungen werden verschiedene Protokolle und Normen verglichen. Des Weiteren wird anhand der Norm IEC 61850 ein standardkonformes Informationsmodell entwickelt, welches die Grundlage für die Kommunikation darstellt. Über ein entsprechendes Mapping auf eine geeignete Übertragungsstruktur wird der Datenaustausch über das Netzwerk ermöglicht.

Anhand der prototypischen Implementierung des Anwendungsfalls zur Einhaltung von Einspeiseprognosen von PV-Anlagen wird im Kapitel 6 eine Evaluierung des zuvor entwickelten Konzepts durchgeführt. Abschließend wird die Arbeit im Kapitel 7 zusammengefasst und ein Ausblick auf weitere mögliche Arbeiten gegeben.

1.4 Ergebnisse

Im Rahmen dieser Arbeit wurden anhand der Norm IEC 61850 Strukturen zur standardkonformen Kommunikation mit einem Batteriespeichersystem entwickelt. Über die Identifikation von Anforderungen an ein solches System sowie die Analyse verschiedener Kommunikationsprotokolle anhand dieser Anforderungen konnte die Norm IEC 61850 als am geeignetesten herausgestellt werden. Die standardisierten Datenstrukturen und Services erlauben die vollständige Steuerung sowie Überwachung des Batteriesystems und bilden neben den reinen Daten ebenfalls die Semantik jener Daten ab.

Über ein von der Norm definiertes Meta Modell und anhand von Konfigurationsdateien können die internen Datenstrukturen flexibel an den entsprechenden Anwendungsfall angepasst werden. Dies, sowie die Realisierung des Anwendungsfalls 'Einspeiseprognose einhalten' konnten anhand des Konzepts erfolgreich umgesetzt werden. Die Eignung der Norm IEC 61850 für die Kommunikation mit Batteriespeichern konnte somit im Rahmen dieser Arbeit gezeigt werden. Insbesondere aufgrund der Flexibilität durch die Konfiguration und der Möglichkeit weitere Datenstrukturen dem Datenmodell hinzuzufügen, ist die Norm äußerst gut für den Informationsaustausch im Smart Grid geeignet.

2 Grundlagen und Stand der Technik

Dieses Kapitel widmet sich den für diese Arbeit wichtigen Grundlagen, die für das weitere Verständnis wesentlich sind. Des Weiteren soll ein kurzer Überblick über den aktuellen Stand der Technik geschaffen werden. Zunächst wird das Konzept des Smart Grids sowie des Kommunikationsstandards IEC 61850 vorgestellt. Außerdem wird der Stand der Technik hinsichtlich weiterer Protokolle für die Kommunikation mit Batteriespeichern sowie aktueller Umsetzungen der Norm IEC 61850 erläutert.

2.1 Das Smart Grid

Derzeit existiert für den Begriff 'Smart Grid' keine einheitliche Definition, da oft verschiedene Aspekte in den Fokus gestellt werden. Gemein ist ihnen, dass es stets um die Integration von erneuerbaren Erzeugungsstrukturen ins bestehende Stromnetz geht [Nau12]. In den letzten Jahren ist ein stetiger Anstieg an Photovoltaik-Anlagen (PV-Anlagen), zumeist bei Privathaushalten, zu verzeichnen. Aber auch der Ausbau von Windparks, insbesondere der der off-shore Windparks, ist stark angestiegen [BS14]. So wird sich auch in der Zukunft die Entwicklung, hin zu einer dezentralen Erzeugungsstruktur, fortsetzten. Hinzu kommt die Volalität der meisten regenerativen Energien. Eine solche Struktur des Netzes, als auch das Ermöglichen des Demand-Side-Response, also gezielte Laststeuerung, machen die Regelung des Stromnetzes und die Fernüberwachung bzw. -steuerung unabdingbar. Somit lässt sich das Smart Grid als ein Stromnetz beschreiben, dass die Integration von erneuerbaren Energien ermöglicht und dabei eine hohe Versorgungssicherheit für die im Netz befindlichen Lasten gewährleistet [Nau12].

Neben dem Streben zu einer flächendeckenden Informations- und Kommunikationstechnologie in der Energiewirtschaft ist auch der Einsatz von verschiedenen Speichertechnologien im Stromnetz der Zukunft von großem Potential. Aufgrund der vielseitigen Einsatzmöglichkeiten im Smart Grid ist insbesondere der Batteriespeicher interessant. Das mögliche Einsatzgebiet erstreckt sich von Übertragungsleitungen zur Spannungshaltung über Fabriken zur Reduzierung von Spitzenlasten, bis hin zum Einsatz in Privathaushalten und dort die Ermöglichung des Inselbetriebs. Folglich ist ein solcher Batteriespeicher allgemein für die Laststeuerung, die in Folge der Volalität unentbehrlich ist, einsetzbar.

Ferner zeichnet sich das Smart Grid durch den Einsatz von sogennanten 'Intelligent Electronic Devices' (IEDs) aus. Dies sind elektronische Bauteile, die neben einer Anbindung an verschiedene Kommunikationstechnologien auch die Möglichkeiten zur lokalen Speicherung von Daten, z.B. Messwerten, besitzen sowie aus einem Mikrokontroller bestehen. Eine Batterie, unabhängig von ihrer Größe bzw. Kapazität, stellt ebenfalls ein IED dar, wenn sie mit entsprechnenden Mikrokontrollern zur Überwachung ausgestattet ist. Das Wort 'Batterie' beschreibt im Kontext dieser Arbeit also das System aus tatsächlichem Speichermedium, entsprechender Überwachungselektronik sowie Hardware zur Netzkommunikation.

Das Smart Grid besteht demnach aus verschiedenen dezentralen, als zum Teil auch stark volatilen, Energieerzeugern, aus steuerbaren IEDs und Kontrollstrukturen sowie

aus Kommunikationstechnologien zum Austausch von Daten und Informationen. Damit der notwendige Informationsaustausch möglich ist, sind Kommunikationsprotokolle, Standards und Normen essentiell. Eine Norm, die für die zukünftigen Aufgaben im Stromnetz entwickelt wurde, ist die Norm IEC 61850.

2.2 IEC 61850

Im Jahr 1995 wurde damit begonnen, die Normenreihe IEC 61850 "Communication Networks and Systems in Substations" zu entwickeln. Diese trat im Jahre 2004 in Kraft und soll proprietäre signalorientierte Protokolle und insbesondere die Norm IEC 60870, in der Stationsautomatisierung ablösen sowie die Interoperabilität zwischen Geräten verschiedener Hersteller verbessern. Signalorientiert bedeutet in diesem Kontext, dass jedes Telegramm einen Datenpunkt repräsentiert. Dieses Telegramm ist über genau eine Adresse sowie einen Datentyp definiert. Über diese Adresse ist dann festgelegt, um welches Signal es sich handelt. Dies bedeutet, dass Sender und Empfänger die Bedeutung dieser Adresse kennen müssen. Die sehr umfangreiche Norm beschreibt nicht nur ein Kommunikationsprotokoll, sondern definiert darüber hinaus außerdem ein generisches, objektorientiertes Datenmodell, eine Beschreibungssprache für IEDs (Intelligent Electronic Devices) sowie weitere Informationsmodelle. Als Besonderheit lässt sich das Datenmodell herausstellen, da hier erstmalig auf Objektorientierung gesetzt wurde und ferner auch die Semantik der jeweiligen Daten beschrieben wird; dies ist bisher bei den verbreiteten Protokollen nicht gegeben (signalorientiert). Es werden somit nicht nur Nummerncodes und Werte übertragen, sondern auch die jeweilige Bedeutung [Tie06]. Hervorzuheben ist jedoch, dass es sich bei dieser Objektorientierung nicht um die Objektorientiertierung, wie sie bei der Objektorientieten Programmierung (OOP) zu finden ist, handelt. Vielmehr werden die Daten in baumartigen Strukturen organisiert, die wiederum in ein baumartiges Datenmodell erzeugen. Tabelle 1 verdeutlicht, durch einen Vergleich von IEC 60870-5-104 und IEC 61850, wie durch Normung die Interoperabilität erreicht und so die Anzahl an Gateways sowie zusätzlicher Entwicklungsaufwand verringert werden soll. Aufgrund der Verbreitung der Normenreihe IEC 60870 wurde der Vergleich mit der Norm IEC 60870-5-104 (Anwendungsbezogene Norm für Fernwirkaufgaben in IP-Netzen) gewählt.

Tabelle 1: Vergleich der Normen IEC 60870-5-104 und IEC 61850 aus [Tie06]

	IEC 60870-5-104	**IEC 61850**
Anwendungsschichten	signalorientiert	Objekte, MMS
Transportschichten	TCP/IP	TCP/IP
Physikalische Schichten	Ethernet	Ethernet
Plattform	Norm	Norm
Datenmodell	Proprietär	Norm
Objekte	Proprietär	Norm

Der Informationsaustausch der Norm IEC 61850 baut hauptsächlich auf einem wohldefinierten Informationsmodell auf. Jenes Modell kann als Kern des IEC 61850 Standards betrachtet werden. Es definiert die Information und die Semantik unabhängig von der konkreten Implementiertung [IEC 61850-7-1]. Von diesem generischen Modell werden je nach Projektierung konkrete Objekte abgeleitet bzw. instanziiert. Die Beschreibung der Objekte und somit der verschiedenen Geräte bzw. IEDs (Intelligent Electronic Devices) erfolgt mit der ebenfalls im Standard definierten Beschreibungssprache SCL (Substation Configuration Language) [IEC 61850-6].

Das Datenmodell ist aus Logical Devices (LD), Logical Nodes (LN), Common Data Classes (CDC) und Attributen aufgebaut. Ein physikalisches Gerät (IED) kann als Server wie auch als Client aggieren. IEDs tauschen Daten aus und rufen Services gegenseitig, je nach Anwendungsgebiet, auf. Die Kommunikation findet somit peer-to-peer statt. Ein IED kann mehrere LDs besitzen, die wiederum aus mehreren LNs betstehen, die die Container für die funktionellen Daten darstellen. LNs kapseln die verschiedenen Funktionen und halten die, je nach Funktion, benötigten Daten. Diese Datenobjekte sind wiederum in CDCs gruppiert, die die unterschiedlichen Attribute halten. Die von einer CDC gehaltenen Attribute sind darüber hinaus in funktionelle Gruppen (functional constraints), wie beispielsweise “Status information (ST)” oder “Configuration (CF)” eingeteilt. CDCs sind insbesondere für die Interoperabilität wichtig, da sie die einzelnen standardisierten Daten sowie Semantik enthalten. Schließlich kapseln die Attribute die tatsächlichen Datenpunkte, Basisdatentypen sowie Settings und ‘Controls’. Die Abbildung 1 veranschaulicht exemplarisch den Aufbau dieses Datenmodells.

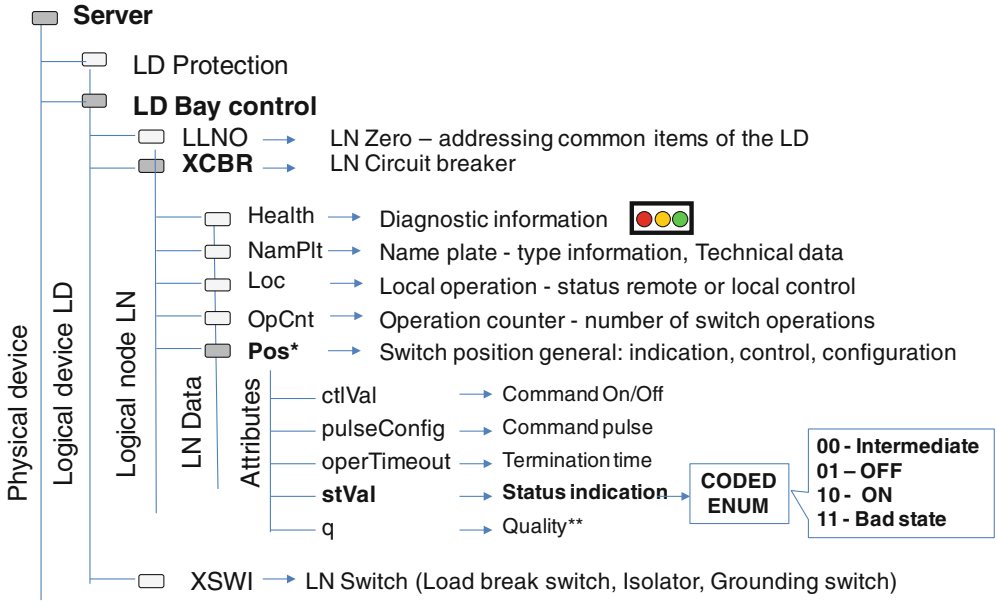

Abbildung 1: Datenobjekte des IEC 61850 Datenmodells (aus [BS14])

Die Abbildung 1 bildet das Schema eines beispielhaften Servers (IED) ab, in dem sich zwei LDs befinden, die für ‘Protection’ sowie ‘Bay control’ zuständig sind (Bay: Teilbereich einer (Umspann-)Station). LD ‘Bay control’ besteht aus drei LNs, welche jeweils eine Funktion im IED kapseln. LLN0 (“Logical node zero”) ist der Standardknoten und in jedem LD vorhanden. Er speichert allgemeine Informationen wie “local

control behavior" oder "Operation time". Logical Node XCBR ("Circuit breaker") ist detaillierter dargestellt. Die einzelnen 'LN Data' und die entsprechenden Attribute sind den CDCs zugeordnet. Das Datenobjekt 'Pos*' ist in diesem Beispiel weiter spezifiziert. Es besteht aus den Attributen 'ctlVal', 'pulseConfig', 'operTimeout', 'stVal' und 'q**' [BS14]. Diese Attribute haben verschiedene Datentypen (Boolean, Coded Enum, String etc.) und stellen damit die letzte Kapselung vor den tatsächlichen Datenpunkten dar.

Der Standard definiert zu den Klassen des Datenmodells eine Menge von Services, die verschiedene Aktionen ausführen können. Über sie lassen sich Werte abfragen oder setzen, Geräte kontrollieren, Dateien hoch- bzw. herunterladen sowie Logs versenden. Das Datenmodell mit den Services sowie unter anderem das Reportmodell und das Loggingmodell machen zusammen das 'Abstract Communication Service Interface' (ACSI) aus. Tabelle 2 zeigt einen Ausschnitt, des vom IEC 61850 definierten ACSI und den insgesamt 55 verschiedenen Services.

Tabelle 2: Ausschnitt: ACSI Services [IEC 61850-7-2]

Server	**Report-Control-Block**
GetServerDirectory	Report
Associate	GetURCBValues
Release	SetURCBValues
Logical Node	**Log-Control-Block**
GetLogicalNodeDirectory	GetLCBValues
GetAllDataValues	SetLCBValues
Data	QueryLogByTime
GetDataValues	QueryLogAfter
SetDataValues	GetLogStatusValues
GetDataDirectory	**Controls**
GetDataDefinition	SelectWithValue
Data-Set	Cancel
GetDataSetValues	Operate
SetDataSetValues	CommandTermination
CreateDataSet	TimeActivatedOperate
DeleteDataSet	**FileTransfer**
GetDataSetDirectory	SetFile
	DeleteFile

Tabelle 2 zeigt unter anderem die Services des Log-Control-Blocks (LCB), des 'Unbuffered Report Controł- Blocks' (URCB), der Data-Sets sowie die Kontroll-Services. Instanzen des URCB werden von dem Logical Node LLN0 gehalten. Jede Instanz eines

*CDC DPC - Common Data Class Controllable Double Point

**Es gibt verschiedene 'Quality' Typen (siehe [IEC 61850-7-3] S. 12)

URCB ist genau einem Client zugeornet. Der URCB ist für das Versenden von Reports zuständig. Das Senden ist dabei von verschiedenen 'Triggeroptionen' abhängig. So kann beispielsweise ein Report an einen Client erfolgen, wenn sich der Wert einer Messung geändert hat. Beim URCB werden Reports direkt versendet, im Gegensatz zum Buffered-Report-Control-Block (BRCB). Hier kann eine Zeit definiert werden, in der Reports gepuffert und anschließend zum Client gesendet werden. Instanzen dieses Blocks gehören ebenfallss zum LLN0. Das Data-Set-Model ermöglicht das Zusammenfassen und Referenzieren von mehreren Datenpunkten. Diese Daten können auf mehrere LNs verteilt sein. Die Kontroll-Services dienen der aktiven Steuerung des IEDs. Es ist möglich, neben der 'direkten Kontrolle' eine 'select before operate'-Kontrolle durchzuführen. Hier wird zunächst geprüft, ob der Client berechtigt ist, eine Statusänderung durchzuführen bzw. ob dies überhaupt möglich ist. Wird auf eine solche Anfrage mit einer positiven Rückmeldung geantwortet, kann vom Client der 'Operate'-Service ausgeführt werden.

Der Standard definiert weiterhin Modelle für "Setting Groups", Generic Substation Events (GSE) und Zeitsynchronisierung. Normalerweise kann ein Datenpunkt nur einen Wert besitzen. Setting Groups ermöglichen es jedoch, zwischen verschiedenen Werten zu wechseln. Dies kann auch gleichzeitig für mehrere Attribute über einen sogenannten 'Setting-Group-Control-Block' erfolgen. Dieser hält eine Reihe an Werten, die editiert werden können, vor, zwischen denen dann gewechselt werden kann. Das 'Generic Subsation Event Model' bietet zwei Nachrichtentypen zum Informationsaustausch in Echtzeit. Mit den GOOSE (Generic Object Oriented Substation Event) Nachrichten können allgemeine Daten, die in Data Sets organisiert sind, versendet werden. Die GSE (Generic Substation State Event) Nachrichten ermöglichen das Versenden von Statusänderungen eines Gerätes. Durch das 'Publisher-Subscriber Model' lassen sich die Daten an mehrer registrierte Clients gleichzeitig per Multi- bzw. Broadcast-Nachricht senden. Ferner ist für das Logging ein 'Log-Control-Block' (LCB) und für das Übertragen größerer Dateien ein 'File Transfer Model' definiert. Wobei der LCB die Logeinträge organisiert und eine Schnittstelle zwischen Client und Log darstellt. Das File Transfer Model' ist einfach gehalten und bietet die Möglichkeiten Daten abzufragen, zu speichern oder zu löschen.

Das ACSI definiert abstrakte, generische Modelle bzw. Klassen (siehe Abb. 2), von denen konkrete Objekte, wie beispielsweise "XCBR", "Pos", "stVal" etc. abgeleitet und instanziiert werden sollen. Dies geschieht über die Beschreibungssprache SCL (Substation Configuration Language). SCL basiert auf XML (Extensible Markup Language) und bietet die Möglichkeit, einzelne IEDs oder sogar ganze Stationen zu beschreiben. SCL beschreibt welche LDs, LNs, CDCs etc. bzw. allgemein, welche Datenobjekte auf einem Gerät (IED) instanziiert werden sollen und somit auch welche Services zur Verfügung stehen.

Abbildung 2 zeigt den Kern des Datenmodells und die jeweiligen Beziehungen der Klassen untereinander. Die dargestellten Klassen sind generisch bzw. abstrakt, das heißt, sie sind im Kern für jedes abgeleitete Objekt gleich. Die jeweilige Identifizierung eines Objekts erfolgt über die Namensgebung bzw. eine spezifische Identifikationsnummer (ID). Beispiele für konkrete Objekte sind rechts über der Klassenbeschreibung gegeben, links befindet sich das respektive SCL-Tag mit dem das entsprechende Date-

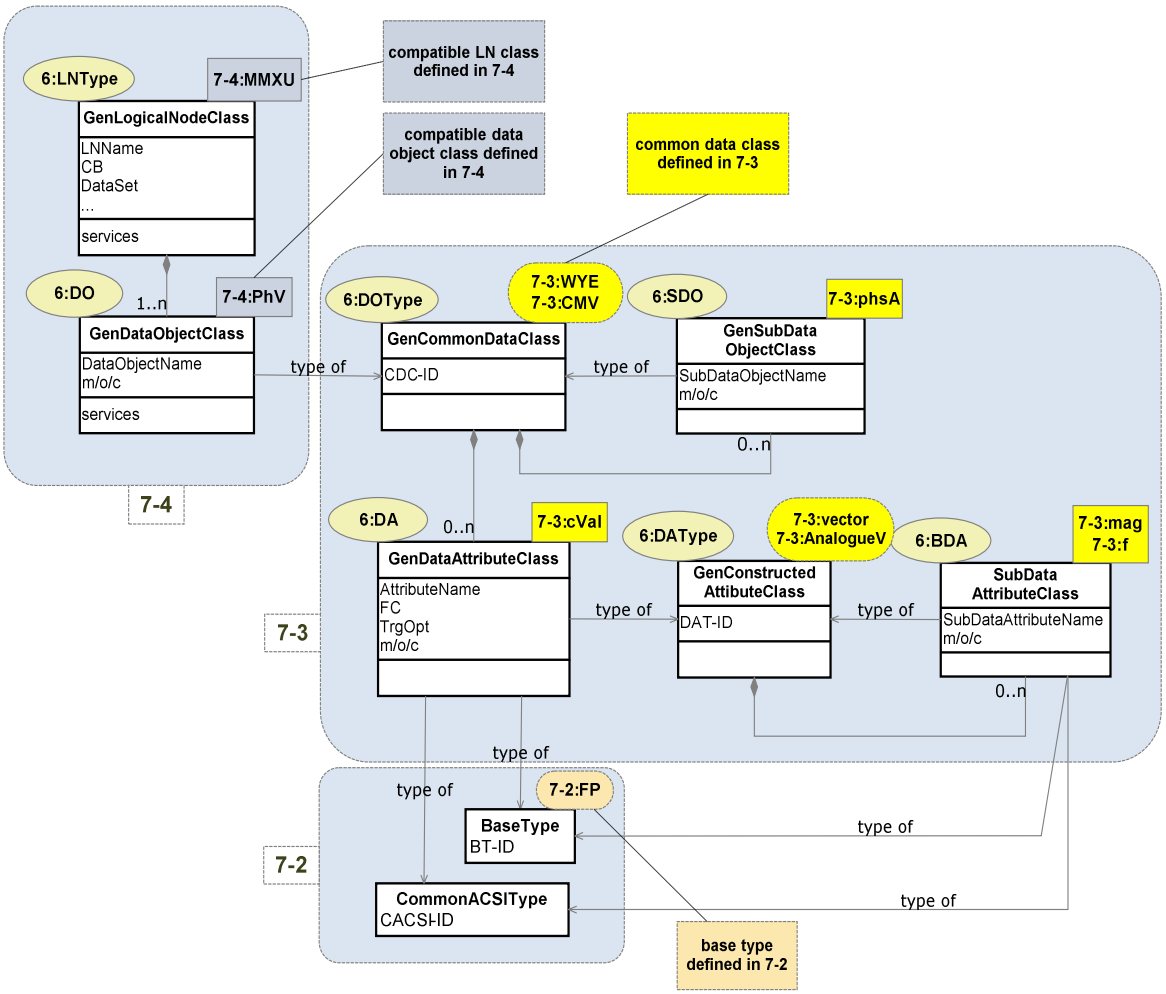

Abbildung 2: Kern des konzeptuellen Meta-Modells [IEC 61850-7-2]

nobjekt in der XML-Struktur identifiziert wird. Die Ziffern geben den zugehörigen Teil des Standards an. Der Standard definiert für jedes LN verschiedene 'DataObjects', wobei diese erneut jeweils verschiedene Attribute besitzen, die in Common Data Classes gruppiert sind. Dazu ist angegeben, ob ein 'DataObject' bzw. Attribut 'mandatory', also obligatorisch, oder 'optional' ist. So kann je nach Projektierung und physikalischem Gerät die passende Auswahl getroffen werden. Diese Auswahl kann dann über eine entsprechend gestaltete SCL-Datei in konkrete Datenobjekte umgesetzt werden. An diesem Schema ist besonders festzuhalten, dass die im Standard beschriebene Objekt Orientierung nicht der Objekt Orientierung des gleichnamigen Programmierparadigmas entspricht. So sind 'kompatible Klassen', wie es auch in den Abbildung 2 zu sehen ist, keine Klassen im Sinne der OOP (Objekt Orientierte Programmierung). Sie entsprechen viel mehr Instanzen, die von der entsprechenden generischen Klasse abgeleitet und je nach Projektierung anhand einer SCL-Datei zusammengesetzt werden. Es entscheidet sich somit erst bei der Instanziierung eines Objektes, wie es tatsächlich aufgebaut ist. Dabei gibt die ID bzw. Namesgebung des Objekts Auskunft über den jeweiligen Typ.

Das ACSI, bestehend aus dem Meta Modell und den verschiedenen Services, bildet

einen Block, der in sich geschlossen ist. Die abstrakten Services stellen eine Schnittstelle zwischen den Datenwerten und dem konkreten Datenaustausch über ein Netzwerk dar. Dies ermöglicht einen relativ einfachen Austausch verschiedener Kommunikationsprotokolle. Die Norm IEC 61850 definiert im Normenteil [IEC 61850-8-1] ein Mapping des ACSI auf das sogenannte MMS (Manufacturing Messaging Specification). Dieser Standard ermöglicht den Austausch der objektorientierten Daten über das Netzwerk. Der Kommunikationsstack ist in Tabelle 3 dargestellt.

Tabelle 3: Schichtenmodell mit Protokollen

Schicht	Protokoll
Anwendung	ACSI
	MMS
Transport	TCP
Netzwerk	IP
Data-Link	Ethernet
Physikalisch	100BASE-T

Über ein Mapping des ACSI auf das MMS kann die Kommunikation über TCP/IP, Ethernet und 100Base-T erfolgen. Dies ist ein Beispielkommunikationsstack, wie er im Standard vorgeschlagen wird und auch in dieser Arbeit Anwendung finden soll. Grundsätzlich ist es jedoch möglich, verschiedene Protokolle für die verschiedenen Schichten zu wählen.

2.3 Stand der Technik

Zur Zeit ist das Stromnetz im Wandel begriffen; man kann jedoch noch nicht von einem Smart Grid sprechen, da der Ausbau von erneuerbaren Energien sowie der damit verbundene Ausbau des Netzes nicht in entsprechender Weise vorangeschritten ist. Demzufolge ist auch der Stand der Technik, in Bezug auf die Kommunikation im Netz sowie insbesondere hinsichtlich der Batteriespeicher, dahingehend einzuordnen.

Zur Zeit ist der wirtschaftliche Einsatz von Großbatteriespeichern noch nicht weit verbreitet. Auch die Norm IEC 61850 ist noch nicht von allen Herstellern von Elektronikbauteilen umgesetzt. Somit wird im Moment die Ansteuerung der Speichersysteme bzw. allgemein der Netzkomponenten oft noch durch proprietäre, herstellerspezifische Softwarelösungen umgesetzt[1]. Der aktuelle relativ geringe wirtschaftliche Einsatz von Batteriespeichersystemen ist außerdem darauf zurückzuführen, dass zur Zeit, insbesondere in Deutschland, die Schwankungen, die durch volatile Energien auftreten noch sehr gut durch deutlich günstigere, flexible Kohle- und Gaskraftwerke sowie Stromimporte aus dem Ausland ausgeglichen werden können [AEW08]. In der Forschung wird jedoch bereits die Kommunikation mit Batteriespeichern über die Norm IEC 61850 simuliert und an Teststänten erprobt[2].

[1] `http://www.ads-tec.de/box/prepare/storaxer-energy-management-system.html`
[2] `http://www.younicos.com/de/technologie/grid_management_software/`

Neben dem Netzausbau und der Forschung an effizienten Speichertechnologien ist insbesondere der umfassende Ausbau an Informations- und Kommunikationstechnologien (IKT) des Stromnetzes wichtig. Vor allem, um die Möglichkeit zur Messung, aber auch der Steuerung von dezentralen Anlagen zu realisieren. Dies ist nicht nur für dezentrale Erzeuger wichtig, sondern ebenso, um Lasten im Netz steuern zu können. Dies ermöglicht neben dem Einsatz im industriellen Umfeld auch die Lastverschiebung durch private Haushalte, denen z.B. über Preissignale ein Anreiz zur Lastverschiebung gegeben werden kann. Insbesondere in diesem Gebiet sowie allgemein zur Wahrung der Netzstabilität kann ebenfalls der Batteriespeicher im Stromnetz Verwendung finden. Dass die flexible Lastverschiebung heute noch nicht umfassend funktioniert liegt vor allem an der zu geringen Abdeckung an IKT. Dass Stromnetz ist also noch nicht ïntelligent", da die Kommunikations-Vernetzung der Komponenten noch nicht in hinreichendem Maße vorangeschritten ist. Bereits wird jedoch von vielen Instituten und Einrichtungen die Forschung in Bereich des Smart Grids vorangetrieben und gefördert [VEU13].[3]

Nicht nur die Norm IEC 61850 wird für den Einsatz im künftigen intelligenten Stromnetz diskutiert. Möglich ist die Überwachung und Steuerung des Netzes ebenso durch die weit verbreitete Norm IEC 60870, mit der heute ein Großteil der Komponenten im deutschen Stromnetz überwacht und gesteuert wird. Insbesondere in Nordamerika wird das Distributed Network Protocol (DNP) 3 für den Datenaustusch in Stromnetzen, aber auch in der Wasserwirtschaft eingesetzt. Weiterhin eignen sich einfach aufgebaute Protokolle für den Echtzeitbetrieb in der Schutztechnik, hier können verschiedenste Feldbusse, wie beispielsweise das EtherCAT oder Profibus zum Einsatz kommen. Darüber hinaus ist das Modbus TCP ein über Jahre bewährstes Kommunikationsprotokoll. Mit ihm lässt sich ebenfalls auf einfach Weise eine Kommunikation mit verschiedenen elektronischen Bauteilen herstellen.

Für die Steuerung und Überwachung eines Batteriesystems eigenen sich grundsätzlich eine Vielzahl an Kommunikationsprotokollen, da diesen Protokollen gemein ist, dass sie dazu entwickelt worden, um Informationen zwischen zwei Systemen auszutauschen. Die einzelnen Protokolle sind jedoch zumeist für einen bestimmten Anwendungsbereich entworfen worden und eigenen sich somit nur noch bedingt für anderweitige Einsatzzwecke. In dieser Arbeit sollen die zuvor genannten Protokolle hinsichtlich ihrer Eignung für Batteriesysteme, aber auch im Kontext des intelligenten Stromnetzes, untersucht werden.

2.4 Umsetzungen von IEC 61850

Die Norm IEC 61850 ist, wie bereits erwähnt, für die Stationsautomatisierung entwickelt worden. In diesem Bereich sind heute die meisten Umsetzungen dieses Kommunikationsstandards zu finden. Heute ist die Norm in der Übertragungsnetzebene und bei der Automatisierung von Schaltanlagen Stand der Technik [SIE14].Hauptsächliches Einsatzgebiet im Übertragungsnetz ist der Informationsaustausch zwischen Leistellen und Umspannstationen. Hier wird über die Fernsteuerung von der Leitstelle aus für

[3] `https://www.web2energy.com`

das Gleichgewicht zwischen Erzeugung und Verbrauch sowie für einen stabilen Systemzustand durch die Netzregelung von Frequenz, Wirkleistung und Spannung gesorgt. Weitere Anwendung findet die Norm im Bereich der Schutztechnik, die durch den Austausch von Daten in Echtzeit dafür sorgt, dass bei Überbelastung gefährdete elektrische Komponenten vom Netz getrennt werden. Der Einsatz in diesem Bereich ist jedoch in Deutschland noch nicht weit verbreitet.

In den letzten Jahren hat sich das Einsatzgebiet der Norm IEC 61850 stets weiterentwickelt und es sind weitere Normenteile erschienen, die den Informationsaustausch zwischen weiteren Komponenten im Stromnetz beschreiben. Hier ist insbesondere im Kontext dieser Arbeit der Normenteil [IEC 61850-7-420] sowie der noch nicht veröffentlichte Normenteil IEC 61850-90-9[4] und zu nennen.

[4]IEC 61850-90-9 – Object Models for Batteries

3 Anforderungsanalyse

Das Kapitel der Anforderungsanalyse beschäftigt sich mit der Ausarbeitung sowie Analyse von Anwendungsfällen bzw. den daraus abgeleiteten Anforderungen. Zunächst werden folgend die Geschäftsanwendungsfälle entworfen, die die für den Anwender wahrnehmbaren Aktionen beschreiben bzw. definieren. Darauf aufbauend werden die zugehörigen Systemanwendungsfälle erarbeitet. Diese definieren und beschreiben die für eine erfolgreiche Kommunikation wichtigen Informationen. Diese Analyse stellt den Grundbaustein für das im folgenden Kapitel entworfene Konzept dar.

3.1 Geschäftsanwendungsfälle

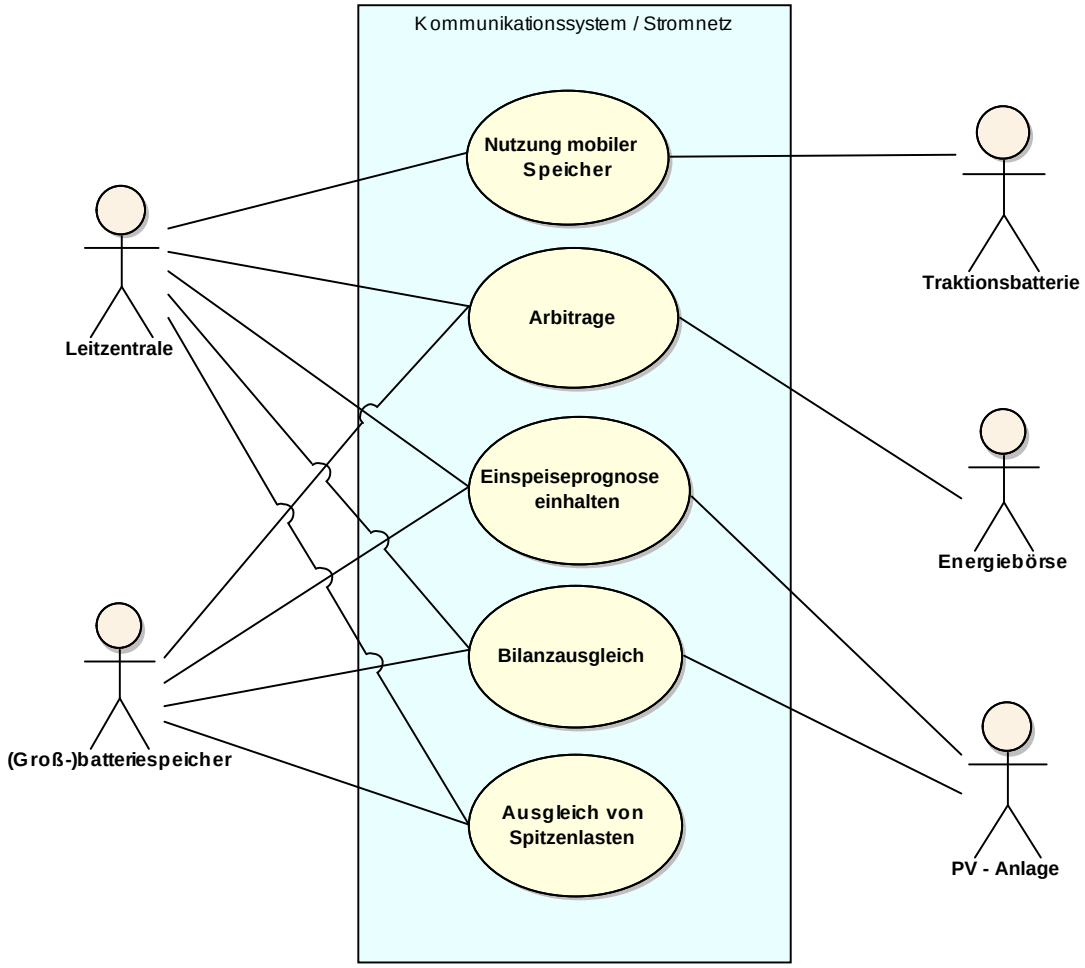

Abbildung 3: UML - Anwendungsfalldiagram: ‘Geschäftsanwendungsfälle’

Abbildung 3 veranschaulicht die im Folgenden näher spezifizierten Geschäftsanwendungsfälle. Zum einen soll die Nutzung mobiler Speicher, hier sind dies Traktionsbatterien von Elektroautos, betrachtet werden. Ein weiterer Anwendungsfall beschreibt die Arbitrage im Zusammenhang mit der zum Teil stark schwankenden Strompreisen und der Möglichkeit, hier Batteriespeicher gewinnbringend einzusetzten. Zwei weitere mögliche Einsatzszenarien für Batteriespeicher ergeben sich aus der Verbindung mit

PV-Anlagen. Ein Batteriesystem kann hier dazu beitragen, eine Einspeiseprognose einzuhalten oder den Leistungsrückfluss, beispielsweise im Verteilnetz, zu vermeiden und so zu einem Bilanzausgleich beizutragen. Überdies soll der Spitzenlastausgleich mittels eines Batteriespeichers bei energieintensiven Produktionen untersucht werden.

3.1.1 Anwendungsfall: Bilanzausgleich

Tabelle 4: Anwendungsfall: Bilanzausgleich

Beschreibung Anwendugsfall	
Name	**Bilanzausgleich**
Kurzbeschreibung	Durch eine hohe Stromeinspeisung von durch PV-Anlagen in das Stromnetz kann die Spannung im Netz erhöht werden und somit ein ungewollter Leistungsrückfluss entstehen. Eine Batterie zwischen "primären Stromerzeuger" und Verbrauchern bzw. beim privaten Erzeuger kann diesen Rückfluss verhindern, indem sie die überschüssige Energie speichert.
Akteure	◇ Leitzentrale ◇ Batterie
Auslöser	Zu hohe Einspeisung und damit verbundene höhere Spannung
Ergebnis	Energie wird bei hoher Produktivität der privaten Erzeuger durch die Batterie gespeichert und bei hoher Last von ihr wieder abgegeben.
Eingehende Daten	Spannung des Stromnetzes, SoC der Batterie
Vorbedingungen	Die Batterie kann geladen werden, SoC ist $< 100\%$
Nachbedingungen	Die Spannung ist stabilisiert
Essenzielle Schritte	1. Überprüfen der aktuellen Spannung durch die Leitzentrale 2. Benachrichtigung der Batterie bei Erreichen des Grenzwertes 3. Bestätigung durch die Batterie 4. Spannungsausgleich durch die Batterie

Immer mehr Hauhalte besitzen private Stromerzeugungsanlagen. Oft in Form einer PV-Anlage. Produziert diese mehr elektrischen Strom, als der Hauhalt verbraucht, kann die überschüssige Energie in das Niederspannungsnetz eingespeist werden. Kommt es zu hoher Einspeisung durch viele Haushalte, kann dies durch eine Spannungserhöhung zu einem unerwünschten Leistungsrückfluss führen. Da das heutige Stromnetz jedoch so konzipiert ist, dass traditionell die Verteilnetzebende über so gut wie keine Kraftwerkseinspeisung verfügt, der Leistungsfluss also stets vom Übertragungsnetz zum Verteilnetz fließt, ist dieser Rückfluss nicht vorgesehen. Dieser ungewollte Fluss kann von einer bzw. mehreren Batterien kompensiert werden, indem sie die überschüssige Energie aufnehmen und so die Spannung stabilisieren. Um dieses Anwendungsszenario umsetzen zu können, müssen zwischen der Batterie und einer Leitzentrale oder Kontrolleinheit gewisse Informationen ausgestauscht werden.

3.1.2 Anwendungsfall: Ausgleich von Spitzenlasten

Tabelle 5: Anwendungsfall: Ausgleich von Spitzenlasten

Beschreibung Anwendugsfall	
Name	**Ausgleich von Spitzenlasten**
Kurzbeschreibung	Damit ein gewisser Stromverbrauch nicht überschritten wird, kann während eine Produktion weniger ausgelastet ist, z. B. nachts, eine Batterie geladen werden. Steigt dann bei hoher Produktion der Stromverbrauch an und droht ein gewisses Niveau zu überschreiten kann die Batterie hinzugeschaltet werden, um so den benötigten Strom aus dem Netz zu senken.
Akteure	◇ Leitzentrale ◇ Batterie
Auslöser	Bei hoher Produktion werden Spitzenlasten erreicht
Ergebnis	Spitzenlasten werden geglättet und es entstehen keine höheren Kosten
Eingehende Daten	Aktueller Stromverbrauch durch das Unternehmen, SoC der Batterie
Vorbedingungen	Durch hohe Produktion wird viel elektrischer Strom benötigt, Batterie kann Strom in entsprechendem Umfang liefern, SoC ist $> 0\%$
Nachbedingungen	Stromspitzen sind geglättet und ein gewisser Stromverbrauch (aus dem Netz) wurde nicht überschritten
Essenzielle Schritte	1. Kontrolle des aktuellen Stromverbrauchs durch eine Leitzentrale 2. Bei hoher Auslastung: zuschalten der Batterie 3. Überwachen des SoC der Batterie

Für energieintensive Unternehmen kann die Minderung von Spitzenlasten lukrativ sein, da sich der Strompreis nicht nur nach der genutzten Energie, sondern auch nach der angeforderten Leistung richtet. Eine Batterie kann hier dazu genutzt werden, um eine gewisse Leistung, die vom Stromanbieter angefordert wird, nicht zu überschreiten. Dazu wird zu Spitzenlastzeiten Energie aus der Batterie angefordert, um so eine vorgegebene Leistungsgrenze nicht zu überschreiten. Zu Zeiten geringerer Produktion bzw. geringerer Stromauslastung kann die Batterie wieder geladen werden. Bei dynamischen Veträgen zwischen Verbraucher und Stromproduzent kann dies insbesondere nachts sinnvoll sein, da dann der Strompreis oft deutlich niedriger ist als am Tag.

3.1.3 Anwendungsfall: Einspeiseprognose einhalten

Tabelle 6: Anwendungsfall: Einspeiseprognose einhalten

Beschreibung Anwendugsfall	
Name	**Einspeiseprognose einhalten**
Kurzbeschreibung	Liegt der tatsächlich erzeugte Strom einer PV-Anlage unter der Prognose so muss Strom von einem anderen Erzeuger produziert werden. Um solche Defizite ausgleichen zu können, kann bei der Erzeugung von mehr Strom als prognostiziert dieser in einer Batterie gespeichert werden. Sinkt die Erzeugung unter die Prognose kann die gespeicherte Energie genutzt werden um dies auszugleichen
Akteure	◇ Leitzentrale ◇ Batterie ◇ PV-Anlage
Auslöser	Speicherung bei hoher Erzeugung, Einspeisung bei zu geringer Erzeugung
Ergebnis	Die Einspeiseprognose wird eingehalten
Eingehende Daten	Aktuelle Stromerzeugung, Aktuell zu liefernder Strom, SoC der Batterie
Vorbedingungen	Profil kann nicht eingehalten werden durch zu geringe Stromerzeugung, Batterie kann Strom in entsprechendem Umfang liefern bzw. Strom kann gespeichert werden, SoC ist $> 0\%$ bzw. $< 100\%$
Nachbedingungen	Die Einspeiseprognose kann eingehalten werden bzw. Strom wird gespeichert
Essenzielle Schritte	1. Kontrolle der aktuellen Einspeisung bzw. Erzeugung 2. Bei zu geringer Erzeugung: zuschalten der Batterie 3. Überwachen des SoC der Batterie

Ein Energieerzeuger, der eine PV-Anlage betreibt, hat dem Netzbertreiber 24 Stunden im Voraus ein Einspeiseprofil vorzulegen, welches einzuhalten ist. Ein Einspeiseprofil gibt an, zu welcher Zeit wie viel Energie von der Anlage ins Netz eingespeist wird. Kann diese Prognose durch zu geringe Erzeugung nicht eingehalten werden, muss der Energieerzeuger Strom von einem anderen Kraftwerk hinzukaufen. Durch eine Batterie kann es möglich sein, dieses Einspeiseprofil einzuhalten, indem die Batterie zu Zeiten hoher Stromerzeugung Energie speichert und bei zu geringer Stromerzeugung diese Energie ins Netz einspeist und so dazu beiträgt, die Prognose zu erfüllen.

3.1.4 Anwendungsfall: Nutzung mobiler Speicher

Tabelle 7: Anwendungsfall: Nutzung mobiler Speicher

Beschreibung Anwendugsfall	
Name	**Nutzung mobiler Speicher**
Kurzbeschreibung	Solange ein Elektroauto zum Laden an eine Stromquelle angeschlossen ist, kann dies von der Leitzentrale wahrgenommen werden und so bei hoher Stromproduktion hier Strom gespeichert werden.
Akteure	◇ Leitzentrale ◇ Traktionsbatterie ◇ Ladesäule
Auslöser	Regenerative Stromerzeuger liefern mehr Strom als aktuell verbraucht wird
Ergebnis	Energie wird bei hoher Produktivität der privaten Erzeuger durch die Batterie gespeichert und bei hoher Last von ihr wieder abgegeben.
Eingehende Daten	Stromeinspeisung der regenerativen Erzeuger, SoC der Batterie, Kapazität der Batterie
Vorbedingungen	Überproduktion an elektrischem Strom, Batterie kann geladen werden, SoC ist $< 100\%$
Nachbedingungen	Elektrischer Strom kann gespeichert werden und regenerative Erzeuger müssen nicht gedrosselt bzw. abgeschaltet werden.
Essenzielle Schritte	1. Kontrolle der aktuellen Stromerzeugung durch eine Leitzentrale 2. Bei Überproduktion: Ausmachen von Batteriestandorten 3. Kommunikation mit der Autobatterie bzw. Ladesäule 4. Speicherung des Stroms

Auch die Batterien von Elektrofahrzeugen haben das Potential, zum Ausgleich von Erzeugung und Last im intelligenten Stromnetz beizutragen. Zur Zeit werden diese Speicher aber ausschließlich als Last im Smart Grid eingesetzt, da die meisten Fahrzeuge noch nicht über die Technik verfügen, um gespeicherte Energie auch wieder ins Netz einspeisen zu können. Ein Anwendungsfall, der sich dieser Idee anschließt, ist das Speichern von überschüssig erzeugter Energie in Batterien von Elektrofahrzeugen. Denkbar ist hier eine Kommunikation zwischen einer Ladesäule bzw. einer Batterie und einer Leitwarte, die registriert, wo und wie viele mobile Speicher im Moment zur Verfügung stehen, so dass je nach Bedarf dort Energie gespeichert werden kann. Solange ein Elektrofahrzeug angeschlossen ist, sollen SoC und Kapazität der Batterie an der Leitzentrale übermittelt werden. Interessant wäre außerdem die Übermittelung eines Zeitplans, der angibt, wie lange der Ladevorgang dauern soll, sodass von der Leitzentrale ein effizienteres Lastmanagement ermittelt werden kann.

3.1.5 Anwendungsfall: Arbitrage

Tabelle 8: Anwendungsfall: Arbitrage

Beschreibung Anwendugsfall	
Name	**Arbitrage**
Kurzbeschreibung	Zu Zeiten in denen in der Wirtschaft weniger Strom gebraucht wird, z. B. nachts, kann der Strompreis deutlich unter dem durchschnittlichen Preisniveau liegen. Kauft man zu diesen Zeiten Strom ein und speichert ihn kann er zu Zeiten höheren Verbrauchs und damit auch höheren Preisniveaus mit Gewinn wieder verkauft werden
Akteure	◇ Leitzentrale ◇ Energie-Börse ◇ Batterie
Auslöser	Strompreis ist niedrig
Ergebnis	Erwirtschaftung eines Gewinns durch günstigen Einkauf und teueren Verkauf
Eingehende Daten	Aktueller Strompreis an der Börse, SoC und Kapazität der Batterie
Vorbedingungen	Preisniveau niedrig genug, Batterie kann geladen werden, SoC ist $< 100\%$
Nachbedingungen	Abnehmer wurde gefunden und durch den Verkauf wird ein Gewinn erzielt
Essenzielle Schritte	1. Kontrolle des aktuellen Strompreises 2. Bei geringem Preisniveau Strom einkaufen und speichern 3. Verkauf des Stroms 4. Bei allen Punkten muss eine Kommunikation mit der Batterie stattfinden und der SoC überwacht werden

Seit einiger Zeit wird auch Strom an extra für diesen Zweck eingerichteten Börsen wie beispielsweise dem EEX (European Energy Exchange) in Leipzig gehandelt. Im Laufe eines Tages kann der Strompreis teilweise sehr stark schwanken. Insbesondere dann, wenn nachts ein großer Überschuss an Energie durch Windanlagen produziert wird. Zu dieser Zeit kann Strom äußerst günstig erworben werden bzw. Verbrauchern von elektrischer Energie wird sogar Geld gezahlt, wenn sie die Überschüssige Energie abnehmen. Aufgrund dieser Preisdifferenzen wäre es also möglich, einen Gewinn zu erwirtschaften, indem Strom günstig eingekauft und wiederum zu Zeiten höherer Preise wieder verkauft wird. Ein Komplex, je nach gewünschter Größe, aus Batterien kann also geladen werden, wenn Strom äußerst günstig ist. Zu Spitzenlastzeiten könnte diese gespeicherte elektrische Energie dann an einen Abnehmer wieder verkauft werden. Neben dem Gewinn kann dieses System zum Lastmanagement genutzt werden bzw. trägt automatisch dazu bei.

3.2 Systemanwendungsfälle

Nach einer Beschreibung der Geschäftsanwendungsfälle sollen nun konkrete Anforderungen an die Kommunikation erarbeitet werden. Dazu ist zunächst zu erörtern, welche Daten zur Verfügung stehen und übertragen werden sollen. Aus diesen Daten ergeben sich somit die Systemanwendungsfälle, welche anschließend den einzelnen Geschäftsanwendungsfällen zuzuordnen sind. Daten, die eine Batterie bereitstellen kann, sind in der Tabelle 9 aufgeführt. Es ist möglich über ein Batteriesystem, das aus mehreren Komponenten besteht, weitere Daten zu erfassen. Im Folgenden soll sich jedoch auf die, für die in dieser Arbeit betrachteten Anwendungsfälle notwendigen, Daten konzentriert werden.

Tabelle 9: Batterieinformationen

Status Informationen	Messwerte	Kontrolle
Modus	Spannung	Batterie an- abschalten
Zustand	Stromstärke	Batterie Test
Status	Ladespannung	Strom einspeisen
Batterietyp	Ladestrom	Strom speichern
Ladestand (SoC)	Kapazität	Wirkleistung
Ladezeit	Blindleistung	Blindleistung
Fahrplan	Wirkleistung	Frequenz
Eigentümer	Nennspannung	
Log - Daten	Frequenz	
Vertragsinformationen	Energie	

Die auszutauschenden Batterieinformationen lassen sich in drei Bereiche aufteilen. Zum einen in die ‘Statusinformationen’; sie beschreiben den Zustand bzw. geben allgemeine Informationen zur Batterie. Daneben kann man noch den Bereich der ‘Messwerte’ unterscheiden. Verschiedene Messaufnehmer zeichnen Messwerte auf, die anschließend im Informationsmodell abgebildet werden und je nach Bedarf dann über ein Protokoll z.B. dem Kontrollsystem zur Verfügung gestellt werden. Darüber hinaus soll eine solche Batterie nicht nur Werte senden, sondern auch Steuerungsbefehle erhalten. Eine Auswahl an diesen Befehlen findet sich im Bereich ‘Kontrolle’. Es soll einer Leitstelle möglich sein, die Batterie an- und abzuschalten, Tests auszuführen sowie zu steuern, ob Strom gespeichert oder ins Netz eingespeist werden soll. Nachdem nun die zu kommunizierenden Informationen betrachtet wurden, soll im Folgenden eine Zuteilung von Systemanwendungsfällen zu jenen Informationen erfolgen. Ferner werden die konkreten Anwendungsfälle genauer erläutert.

Beim *Abfragen von Messwerten* soll nicht nur der tatsächlich abgetastete Wert kommuniziert, sondern hinzu sollen weitere Meta - Informationen übertragen werden,

Tabelle 10: Zuordnung: Systemanwendungsfälle zu Informationen

Systemanwendungsfall (Kommunikation)	**Informationen**
Messwerte bzw. Status abfragen	Messwert- und Statusinformationen (Tabelle 9)
Sollwerte vorgeben / Kontrollbefehl senden	Blind- und Wirkleistung, Ladestrom und -spannung, Kontrollbefehle (Tabelle 9), Modus
Fahrplan senden	Schedulingplan
Report bei Wertänderung	Name des Datenpunkts und Wert
Datei senden	Log-, oder Konfigurationsdatei

welche jenen Messwert weiter spezifizieren. Hier ist insbesondere die Einheit des Messwertes zu nennen. Sie, wie auch die Qualität der Messung, sind wichtig, um richtige und zuverlässige Aussagen zu treffen bzw. weiterführende Berechnungen durchzuführen. Ein Name oder eine Adresse für den Wert sind wichtig, um ihn im richtigen Kontext zu interpretieren und feststellen zu können, an welcher Stelle im System die Messung vorgenommen wurde. Weiterhin ist die zusätzliche Übertragung eines Zeitstempels relevant, um die Messung auch zeitlich einordnen zu können. Eine Beschreibung oder Definition kann abschließend weitere Informationen zur Messung bzw. zum Messwert beitragen.

Neben dem *Sollwert*, der übertragen werden soll, muss auch ein Kontrollbefehl näher beschrieben werden, um einen sicheren Betrieb gewährleisten zu können. Für einen Befehl als auch einen Sollwert ist die Adresse, an welche die Daten geschickt und gespeichert werden sollen, essentiell. Der konkrete Befehlsempfänger muss also eindeutig definierbar sein. Daneben ist eine Rückmeldung vom System, je nach Erfolg der Speicherung des Wertes oder der Umsetzung des Befehls, sinnvoll.

Ein, gerade für eine Batterie wichtiger Systemanwendungsfall, ist das Verfolgen und Abgleichen von *Fahrplänen*, also die zeitliche Vorgabe, wann wie viel Strom einzuspeisen ist bzw. ab welcher höheren Erzeugung Strom gespeichert werden kann. Wichtig ist auch hier, dass gewisse Meta-Informationen über den Fahrplan vorhanden sind, damit er richtig ausgelesen bzw. interpretiert werden kann. Zu diesen Informationen zählt zum einen die Angabe der Leistung pro Zeit sowie zu beiden Werten jeweils die Einheit. Weiterhin sind Wirk- und Blindleistung wie auch die Spannung anzugeben.

Damit eine erfolgreiche *Wertänderung*, die durch entweder das Leitsystem oder das interne Batteriesystem veranlasst wurde, wahrnehmbar ist, ist es notwendig, bei entsprechenden Ereignissen eine Meldung bzw. Report zu versenden. Dies kann die Quittierung eines erhaltenen Befehls bzw. Wertes sein, aber auch eine Art Update für die Veränderung beispielsweise der anliegenden Ladespannung sein. Auch beim Senden dieser Daten sind einige weitere Informationen wichtig, um die erhaltenen Daten richtig zu verarbeiten. Hier ist die Einheit des Werts oder der Name des Befehls, eine

Adresse sowie ein Zeitstempel zur chronologischen Einordung als weitere Informationen mitanzugeben.

Desweiteren ist das *Senden einer Konfigurationsdatei* essentiell, da ein IED stets durch eine solche Datei neu konfigurierbar sein soll. Ähnlich dem Senden eines Fahrplans ist darauf zu achten, dass diese Dateien relativ (zu den anderen Systemanwendungsfällen) viele Daten enthalten und so bei der Versendung unter Umständen eine Stückelung in Datenfragmente nötig ist. Neben den Systemanwendungsfällen, die für die Kommunikation, also die Übertragung von Daten zuständig sind, sind weitere Anforderungen an das System aus den Anwendungsfällen ableitbar. Zum einen ist es wichtig, dass ein lokales Ereignisprotokoll (Log) geführt wird, um vergangene Vorgänge beispielsweise bei einer Wartung oder beim Auswerten eines Fehlers nachvollziehen zu können. Dazu muss ein entsprechender Speicher sowie Kontrollstrukturen zur Verfügung stehen. Darüber hinaus muss der Systemanwendungsfall 'Datei senden' dahingehend erweitert werden. Weiterhin erfordert die *(Neu-)Konfiguration* des Systems nicht nur eine Übertragung der Konfigurationsdatei. Es bedarf zusätzlich der lokalen Umsetzung dieser Vorgabe durch entsprechende Strukturen. Diese Umsetzung ist somit als weitere Anforderung an das System gegeben. Tabelle 11 gibt eine Übersicht der Zuordnung zwischen Geschäfts- und Systemanwendungsfällen.

Tabelle 11: Zuordnung: Geschäfts- zu Systemanwendungsfällen

Geschäftsanwendungsfall	Systemanwendungsfälle
Nutzung mobiler Speicher	Messwerte/ Status abfragen, Fahrplan senden, Report bei Wertänderung
Arbitrage	Messwerte/Status abfragen, Kontrollbefehl senden, Report bei Wertänderung, Datei senden
Einspeiseprognose einhalten	Messwerte/Status abfragen, Kontrollbefehl senden, Fahrplan senden, Datei senden
Bilanzausgleich	Messwerte/Status abfragen, Kontrollbefehl senden, Report bei Wertänderung, Datei senden
Ausgleich von Spitzenlasten	Messwerte/Status abfragen, Kontrollbefehl senden, Report bei Wertänderung, Datei senden

Um die Geschäfts- bzw. Systemanwendungsfälle umsetzen zu können, müssen ebenfalls gewisse *nicht funktionale Anforderungen* erfüllt sein. Zum einen ist es wichtig, dass die Datenpakete routbar sind. Die Pakete sollen größere Distanzen durch vermaschte Netze, beispielsweise das Internet, zurücklegen. Zum anderen ist in der Energiewirtschaft, insbesondere im Zuge der Wandlung hin zum 'intelligenten Netz', die Anforderung an die Interoperabilität der verschiedenen elektronischen Geräte erhöht. Immer mehr Geräte sollen ohne große Komplikationen zusammen arbeiten können. Immanenter Bestandteil der Ermöglichung der Interoparabilität ist die Normung von Kommunikationsprotokollen, Schnittstellen und Datenmodellen. Daneben kann als weitere Anforderung an die zu entwickelnden Softwarestrukturen die Kapselung genannt

werden. Die Kapselung bestimmter Softwareteile sowie die Bereitstellung entsprechender Schnittstellen ermöglicht das Austauschen dieser Module. So kann z. B. gut auf die sich schnelle Entwicklung von Informationstechnologien und Kommunikationsprotokollen reagiert werden. Der Informationsaustausch zwischen verschiedenen elektronischen Bauteilen und der Batterie innerhalb einer Station erfordert oft, insbesondere in der Schutztechnik, harte Echtzeit, um z.B. das schnelle Auslösen von Schaltungen zu gewährleisten. Diese Kommunikation ist nicht Teil dieser Arbeit und wird daher in den weiteren Betrachtung vernachlässigt.

3.3 Zusammenfassung

In diesem Kapitel wurden verschiedene Geschäftsanwendungsfälle erläutert. Es wurde erarbeitet in welchen Szenarien der Einsatz von Batteriesystemen interessant ist sowie entsprechende Akteure, Bedingungen und essentielle Schritte für die einzelnen Anwendungsfälle identifiziert. Die Geschäftsanwendungsfälle stellen den Anwendungsfall dar, ohne zu spezialisiert zu sein und ermöglichen jedem Anwender das Verständnis und die kritische Beurteilung des Sachverhalts.

Aus diesen Geschäftsanwendungsfällen wurden im Anschluss spezielle, batteriespezifische Systemanwendunugsfälle erarbeitet, aus denen dann die konkreten Anforderungen an das Batteriesystem abgeleitet wurden. Die abgeleiteten Anforderungen an das System sind die (Mess-)Wertübertragung, das Vorgeben von Befehlen bzw. Sollwerten, das Versenden von Reports sowie von Fahrplänen, die Möglichkeit des Loggings sowie die der (Neu-)Konfiguration des Geräts. Daneben sind weitere nichtfunktionale Anforderungen identifiziert worden (siehe Tabelle 12). Ferner konnte eine Übersicht über die Batterieinformationen, die für einen Betrieb des Batteriesystems notwendig sind, gegeben werden. Folgend schloss sich eine Zuordnung von Informationen, Systemanwendungsfällen sowie Geschäftsanwendungsfällen an. Somit konnte eine Verknüpfung zwischen Daten, Anforderungen und Anwendugsfällen hergestellt werden.

An das Kapitel der Anforderungsanalyse soll sich nun im Konzeptkapitel mit der Entwicklung eines Konzeptes zur Umsetzung der Anwendungsfälle auseinandergesetzt werden.

4 Konzept

In diesem Kapitel soll zunächst auf der Grundlage der zuvor durchgeführten Anforderungsanalyse eine Analyse sowie Bewertung möglicher Lösungsansätze erfolgen, an die sich die Entwicklung eines Konzepts für die Kommunikation mit Batteriespeichern anschließt. Das Konzept umfasst die Erarbeitung eines Informationsmodells, das neben einem Datenschema ebenso Schnittstellen und Kontrollstrukturen zum Datenaustausch und zur Steuerung einer Batterie, bereitstellt.

4.1 Analyse möglicher Lösungensansätze

Die bereits in den Grundlagen genannten Bussysteme und Protokolle sollen in diesem Abschnitt auf ihre grundsätzliche Nutzbarkeit für Batteriesysteme auf theoretischer Ebene untersucht werden. Dies soll unter anderem anhand der zuvor erarbeiteten Systemanwendungsfälle geschehen.

Das *Distributed Network Protocol* (DNP)3 ist ein, insbesondere in Nordamerika, weit verbreitetes Protokoll in der Fernwirktechnik. Es baut in neueren Versionen ebenfalls auf dem Internetprotokoll TCP/IP bzw. UDP/IP auf und wird als Übertragungsprotokoll zwischen Leitstellen und Unterstationen, vorwiegend in der Energie- und Wasserwirtschaft, eingesetzt. Hier regelt es den Datenaustausch zwischen einem SCADA-Master (Supervisory Control and Data Acquisition) und einer Remote Terminal Unit oder einem IED. Das Protokoll baut auf dem Enhanced Performance Architecture (EPA) Modell auf, das aus Gründen der Performanz lediglich aus der Data-Link-Schicht, der Transportschicht sowie der Anwendungsschicht des ISO/OSI Models besteht [RMW04]. Das DNP3 Protokoll ähnelt stark dem des IEC 60870-5 Standards und ist darüber hinaus insgesamt robuster und effizienter als ältere Protokolle, wie z. B. Modbus.

Grundsätzlich könnten die definierten Systemanwendungsfälle durch das DNP3 Protokoll umgesetzt werden, hier müsste jedoch auf eine ‘Logging Funktion’ verzichtet werden, da diese in jenem Protokoll nicht vorgesehen ist. Aus [DNP11] geht außerdem hervor, dass DNP3 im Anwendungsbereich von verteilten Speicheranlagen nicht alle gewünschten Funktionen abbilden kann. Dies gilt insbesondere für die Kontrolle im Inselbetrieb sowie das bereits genannte Auslesen von lokal gespeicherten Logs und Reports, hier kann der DNP3 Standard keine Anforderung erfüllen. Auch weitere relevante Funktionen wie beispielsweise das Spitzenlast-Management oder die Frequenzregulation kann der DNP3 Standard nicht vollständig bedienen. Des Weiteren konnte in [OSS13] gezeigt werden, dass bei hoher Netzwerkauslastung in einem heterogenen Netzwerk das Protokoll, aufgrund von zu hohen Paketverlusten, kaum bzw. nicht mehr praktikabel einsetzbar ist. Ferner ist DNP3 nicht dafür vorgesehen, sämtliche zukünftige Funktionen und Aufgaben in einem Smart Grid zu übernehmen und ist daher für diesen Kontext unpassend [DNP12].

Aufgrund der starken Ähnlichkeit von *Feldbussen*, hinsichtlich der betrachteten Anwendungsfälle, sollen hier unter diesem Begriff exemplarisch zwei Ausführungen betrachtet werden. Zum einen das EtherCAT (Ethernet for Control Automation Technology) Protokoll sowie der Profibus (Process Field Bus). Bussysteme, deren Entwicklung

und Funktionsweise nicht für die Automatisierung von Energiesystemen oder die Prozessautomatisierung ausgelegt sind, sollen in dieser Betrachtung vernachlässigt werden. Feldbusse dienen in der digitalen Übertragungstechnik der Kommunikation zwischen einem Steuergerät und verschiedenen Messaufnehmern bzw. Stellgliedern. Zumeist wird dabei das Master/Slave Modell verwendet, wobei das Steuergerät den 'Master' darstellt und die verschiedenen Feldgeräte die 'Slaves'. Um bereits vorhandene Netzwerke nutzen zu können, findet man bei vielen Feldbussen mittlerweile eine (Echtzeit-)Ethernet sowie TCP/IP Erweiterung [SW06].

Das EtherCAT ist ein solcher Standard, der auf dem Echtzeit-Ethernet und TCP/IP aufbaut bzw. aufbauen kann. Für Echtzeitdaten werden jedoch nur die drei EPA Schichten implementiert. EtherCAT zeichnet sich dadurch aus, dass Datenströme nicht nur an einen Empfänger, der die Daten aufnimmt und interpretiert, gesendet werden. Ein Telegram durchläuft einen Slave, wobei Daten ein- bzw. ausgelesen werden und wird dann weiter geschickt, um vom nächsten Teilnehmer bearbeitet zu werden. Dabei wird ein solches Telegram zunächst nicht komplett empfangen, sondern die Verarbeitung beginnt frühst möglich, nachdem genug Informationen empfangen wurden. Dies gilt ebenso auch für das Versenden. Ein weiterer Geschwindigkeitsgewinn wird dadurch erreicht, dass die gesamte Protokollbearbeitung in Hardware stattfindet [JB03].

Profibus ist ein weit verbreiteter Feldbus in der Fertigungs- und Prozessautomatisierung, welcher nicht auf (Echtzeit-)Ethernet aufbaut. Man unterscheidet zwei Varianten: Profibus DP und Profibus PA. Profibus DP (Dezentrale Peripherie) dient der Ansteuerung von Sensoren und Aktoren durch einen zentralen Master und zeichnet sich durch einen schnellen zyklischen Datenaustausch zwischen jenem Master und dem Slave aus. Profibus PA (Prozess-Automation) beschreibt die Kommunikation zwischen Mess- und Prozessgeräten. Die Busleitung kann hier ebenfalls die Energieversorgung des Feldgerätes übernehmen und ist somit geeignet, um in explosionsgefährlichen Bereichen eingesetzt zu werden. Profibus wird speziell im Bereich der Sensorik und Aktorik eingesetzt, da hier besondere Anforderungen an die Echtzeit, und damit an die Übertragungsgeschwindigkeit von Daten, gestellt werden [SW06].

Sowohl EtherCAT als auch Profibus sind hauptsächlich für die Echtzeit-Kommunikation zwischen Feld- und Steuerungsgeräten ausgelegt. Der Fokus liegt auf kurzen Buszykluszeiten sowie auf berechenbaren maximalen Reaktionszeiten der Geräte [SW06]. Dieser einseitige Fokus macht einen Feldbus für die in dieser Arbeit betrachteten Szenarien unbrauchbar, da Feldbusse für eine Kommunikation zwischen Messfühlern bzw. Stellgliedern und einem Steuerungsgerät konzipiert sind und nicht für eine Kommunukation zwischen einer IED und einer Leitstelle.

Das *Modbus TCP* Protokoll ist ein in der Applikationsschicht arbeitendes Protokoll, welches das TCP in der Transportschicht des ISO/OSI Models verwendet. Es hat sich seit 1979 zu einem 'de-facto Industrie-Standard' entwickelt und bietet simple client/server Kommunikation an. Der Aufbau einer Modbus Nachricht ist äußerst einfach gehalten. Er besteht aus einer Client- bzw. Serveradresse, einem Function code, der angibt welche Operation durchzuführen ist, einem Datenblock sowie einem Error Check. Adressiert werden können stets nur einzelne Bits oder mehrere zwei Byte große Register. Diese können einzeln oder im Block, also mehrere in einer Nachricht, gelesen bzw. beschrieben werden [RMW04].

Aufgrund der Einfachheit des Protokolls können komplexere Aufgaben, beispielsweise das Versenden von Dateien oder 'Logging', nur mit viel Aufwand und äußerst umständlich gelöst werden. Dies liegt unter anderem daran, dass Modbus nur einfache Byteströme versendet, die auf der Empfängerseite zunächst interpretiert werden müssen. Eine Übertragung von Sollwerten, einfachen Kontrollbefehlen und die Abfrage von Messpunkten ist möglich, jedoch fehlt bei der Übertragung stets die Semantik der übertragenden Daten, diese muss an anderer Stelle bereits bekannt sein. Außerdem ist eine eventbasierte Kommunikation, beispielsweise die Benachrichtigung bei der Änderung von Messwerten, nicht möglich. Somit muss das Modbus TCP Protokoll, für die in dieser Arbeit beschriebenen, Anwendungsfälle als ungeeigent angesehen werden [RMW04].

Die *IEC 60870* Norm wurde als einheitliches Protokoll für die Schutz- und Leittechnik entwickelt, um die Interoperabilität zwischen Geräten verschiedener Hersteller zu erhöhen. Vornehmlich definiert der IEC 60870-5-101 Standard Attribute der zu übertragenden Daten sowie ihre jeweilige Kodierung [VDE08]. Die Bedeutung des Inhalts ist jedoch weiterhin seperat allen Kommunikationsteilnehmern zu vermitteln. Aus diesem Grund entstanden über die Zeit viele herstellerspezifische Datenmodelle, die erneut die Interoperabilität erschwerten bzw. sie nur auf einer niedrigen Ebene gewährleisten konnten. Weiterhin kann eine einmal bei der Projektierung eingesetzte Systemkonfiguration nur schwer neu konfiguriert werden, da der Inhalt fest definierten Datenfeldern zugeordnet ist [Nau12].

Da die Datenmodelle im IEC 60870 Standard nicht standardisiert und auch Neukonfigurationen erschwert sind, so ist die Flexibilität eines solchen Standards nicht ausreichend, um künftigen Anforderungen an die Kommunikation gerecht zu werden [VDE08]. Der IEC 61850 Standard geht über ein reines Netzwerkprotokoll hinaus und bietet neben einer Konfigurationssprache, zum flexiblen Modellieren von Geräten sowie ganzen Stationen, außerdem eine abstrake Datenmodellbeschreibung, die auf verschiedene Kommunikationsprotokolle abgebildet werden kann.

Die Norm *IEC 61850* definiert vollumfänglich die Kommunikation, also den Datenaustausch zwischen IDEs. Wie in den Grundlagen gezeigt wurde, stellt eine Batterie, die mit entsprechnder Peripherie ausgestattet ist, ebenfalls eine IED dar. Somit ist die Anwendung des Standards IEC 61850 grundsätzlich möglich. Essentiell sind jedoch die logischen Strukturen und Schnittstellen, die ein Standard zur Verfügung stellen muss, um angemessen die bereits definierten Anwendungsfälle umzusetzen.

Das, dem Standard zugrunde liegende, Konzept kann auch auf eine Batterie angewendet werden, da das Konzept zunächst unabhängig von der Art des IED ist. Neben Kontrollblöcken, die eine lokale Logging-Funktion ermöglichen, ist ein 'File transfer model' definiert, so dass auch Dateien übertragen werden können. Obwohl das Konzept des Standards zunächst für die Stationsautomatisierung entwickelt wurde, und somit die Datenmodelle, Common Data Classes, Logical Nodes und das ACSI hauptsächlich Definitionen für Geräte aus diesem Bereich bieten, ist es doch, insbesondere in der Edition 2.0, auch für die in dieser Arbeit betrachteten Anwendugsfälle einsetzbar.

Um die Semantik der Datenpunkte für eine Batterie standardkonform zu beschreiben, empfielt sich der Normenteil IEC 61850-7-420. Er definiert LNs für die dezentrale

Energieversorgung. Dazu zählen unter anderem zwei LNs für Batteriesysteme: ZBAT (battery system) und ZBTC (battery charger). Diese Datenstrukturen halten einen Großteil der, für die betrachteten Anwendungsfälle notwendigen, Daten, wie beispielsweise Statusinformationenen, Nennspannung, Kapazität, Frequenz etc.. Neben diesen, für ein Batteriesystem spezialisierten, LNs sind weitere logische Knoten zum Speichern von allgemeinen Informationen, zum Aufnehmen von Messwerten und zur Behandlung von Alarmen notwendig. Die Norm IEC 61850 definiert für jeden dieser Bereiche geeignete logische Knoten, wie beispielsweise MMXU für verschiedene Messwerte.

Das Abfragen von Messwerten wird über die vom ACSI definierten Schnittstellen realisiert bzw. erfolgt durch das Auslösen von Triggern über das URCB bzw. BRCB. Des Weiteren kann die Vorgabe von Sollwerten, oder das Steuern bzw. Kontrollieren der Batterie über die definierten Kontrollblöcke geschehen. Ferner ist es möglich für das Einspeisen bzw. Laden Schedulingpläne in den LNs ZBAT und ZBTC zu hinterlegen. Das Speichern eines State-of-Charge (SoC) ist in beiden LNs jedoch nicht möglich. Aufgrund der Wichtigkeit dieses Datenwertes muss dieser dem Datenmodell hinzugefügt werden, sodass es möglich ist die Semantik sowie den Wert abbilden zu können. Es ist möglich, nach den strengen Vorgaben der Norm, weitere standardkonforme Datenstrukturen zu definieren.

4.2 Bewertung möglicher Lösungsansätze

Es kann festgehalten werden, dass der Großteil der betrachteten Strukturen eine Umsetzung der Anwendungsfälle erschwert bzw. nicht möglich gemacht hätte. Das Prinzip eines Feldbusses stellt sich dabei als am wenigsten brauchbar heraus, da es nicht für dezentrale, stärker vermaschte Netze (Wide Area Network) konzipiert ist (siehe dazu auch Abb. 6). Neben DNP3 besitzt auch die Norm IEC 61870 eine Erweiterung des Protokolls (IEC 60870-5-104), so dass die Kommunikation über Ethernet sowie TCP/IP stattfinden kann und so die Möglichkeit besteht, auch über ein WAN (Wide Area Network) Daten auszutauschen. Beiden Protokollen fehlt jedoch ein Datenmodell, das ebenfalls die Semantik der übertragenden Daten beschreibt. So gibt es bei der Norm IEC 60870 zwar klar definierte Datenfelder, diese können jedoch je nach Nutzer bzw. Hersteller unterschiedlich genutzt werden, so kann eine Interoperabilität nicht effizient umgesetzt werden. Modbus TCP basiert ebenfalls auf Ethernet sowie auf TCP/IP ist jedoch, ähnlich den Feldbussen, für komplexere Aufgaben, wie es die erarbeiteten Anwendugsfälle sind, ungeeignet.

Tabelle 12 veranschaulicht abschließend die vorangegangene Analyse. Die verschiedenen Anforderungen an das System sind als Zeilen aufgeführt. Die entsprechenden Zeichen geben an, wie gut bzw. schlecht sich die einzelnen Aspekte von der Norm bzw. von dem Protokoll umsetzbar sind. Die Norm IEC 61850 erfüllt die Anforderungen, die die Anwendugsfälle stellen, am besten. Für die einzelnen Systemanwendungsfälle gibt es spezialisierte, wohldefinierte Datenmodelle und Schnittstellen, die jedoch zum Teil, insbesondere bei der Anforderung der ‘Konfiguration’, einer Erweiterung bedürfen. Des Weiteren bildet jene Norm neben den einfachen Daten ebenso auch die jeweilige Bedeutung jener Daten ab, so dass bei einer Übertragung die anschließende Interpretation einer Nachricht bzw. eines Telegramms entfällt. Ferner ist die Norm aus Gründen der

Tabelle 12: Vergleich: Kommunikationsprotokolle

Anforderung	Feldbus	DNP3	Modbus TCP	IEC 60870	IEC 61850
(Mess-)Wertübertragung	++	++	++	++	++
Reports	−−	++	−−	++	++
Fahrplan senden	∘	+	∘	∘	++
Konfiguration	−−	−−	−−	−−	+
Logging	−−	−−	−−	∘	++
Nichtfunktionale Anf.	−	∘	∘	∘	++

++ 'voll umsetzbar' −− 'nicht umsetzbar'
+ 'umsetzbar' − 'schwer/eher nicht umsetzbar'
∘ umständlich/nicht zielführend umsetzbar

Interoperabilität entwickelt worden und stellt damit sowie mit ihrem Konzept eine zukunftssichere Lösung dar.

Da in den Dokumenten der Norm IEC 61850 kaum Bezug auf eine programmiertechnische Umsetzung genommen wird, ist außerdem eine gewisse Anpassung des Informationsmodells, wie es im Standard definiert wird, wichtig, damit eine möglichst modellgetriebene Softwareentwicklung und effiziente Umsetzung möglich ist. Des Weiteren sind für den batteriespezifischen Einsatz verschiedene Strukturen dem Standard hinzuzufügen. Folgend werden jene Teile des Informationsmodells identifiziert, die für die Erfüllung der Anforderungen notwendig sind. Daran schließt sich die Entwicklung des konkreten Datenschemas sowie ein geeignetes Mapping auf einen entsprechenden Kommunikationsstandard an. Abschließend soll erarbeitet werden, wie konkrete Datenobjekte anhand einer Konfigurationsdatei sowie anhand des entwickelten Datenschemas instanziiert werden können.

4.3 Das Meta-Informationsmodell

In den Grundlagen wurde bereits ein grober Überblick über die Strukturen des IEC 61850 Standards gegeben. Abbildung 2 im Kapitel 2 zeigt das Meta-Modell, wie es vom Standard beschrieben wird. Es zeigt die Verbindungen der grundlegenden Datenstrukturen, auf denen das Informationsmodell des Standards aufbaut. Da es sich bei diesem Schema nicht um ein streng UML-konformes Klassendiagramm handelt, ist eine Überstzung in ein UML-Klassendiagramm zunächst sinnvoll, insbesondere im Hinblick auf eine mögliche Codegenerierung, als auch im Sinne einer modellbasierten Softwareentwicklung. Problematisch sind in der Abbildung 2 die nicht näher spezifizierten gerichteten 'type of' Assoziationen. Dies zeigt sich insbesondere bei den Verbindungen zwischen 'GenCommentDataClass' und 'GenSubDataObjectClass' sowie zwischen 'GenConstructedAttributeClass' und 'SubDataAttributeClass'. Hier ist sowohl eine Aggregation als auch eine 'type of' Assoziation angegeben.

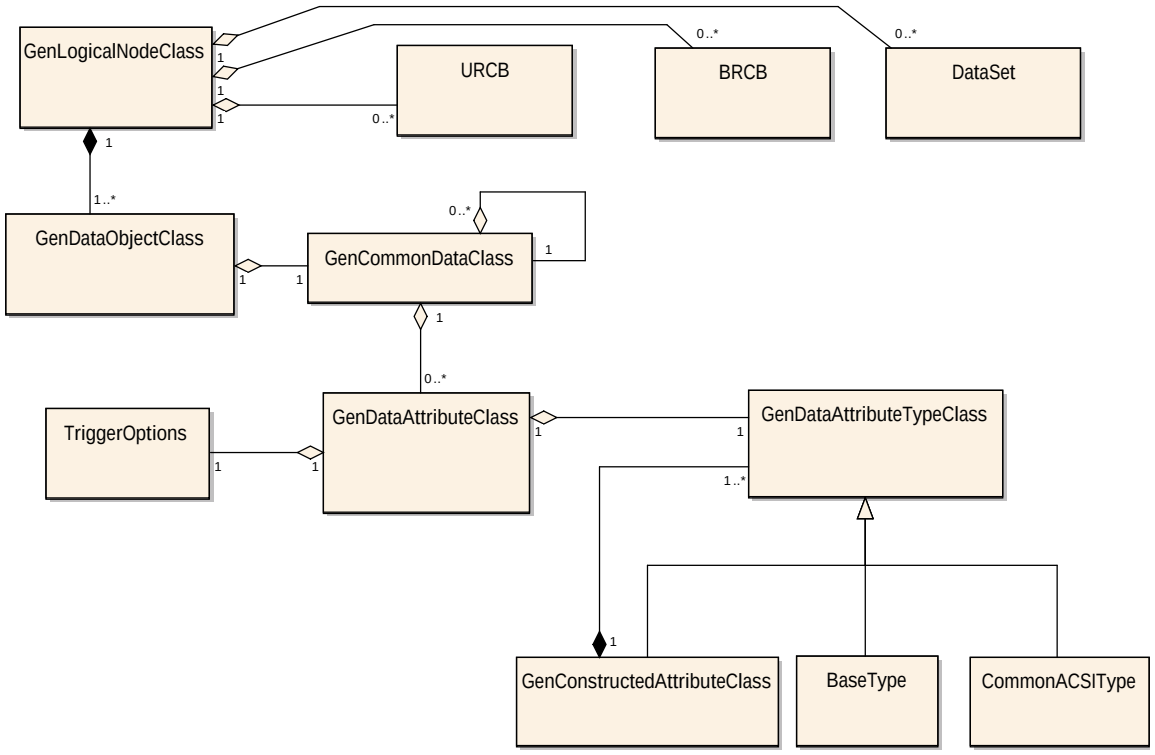

Abbildung 4: UML - Meta Modell in Anlehnung an Abb. 2

Abbildung 4 zeigt das aus Abbildung 2 abgeleitete Klassendiagramm. Es ist der Übersicht halber ohne Attribute und Operationen dargestellt. Das detaillierte Klassendiagramm ist in Anhang A.1 zu finden. Die 'type of' Assoziationen wurden hier durch Aggregationen ersetzt. 'GenSubDataObjectClass' erhält eine Aggregation auf sich selbst, da es möglich ist, dass ein DataObject eine Liste weiterer DataObjects enthält. 'GenCommonDataClass' kann wiederum Objekte dieser Klasse enthalten. Des Weiteren wurde zum Modell eine Hilfsklasse 'GenDataAttributeTypeClass' hinzugefügt. Sie ist für eine spätere Implementierung interessant, da sie eine Abstraktionsebene für die Klassen 'GenConstructedAttributeClass', 'BaseType' und 'CommonACSIType' darstellt und so Instanzen dieser Klasse als Liste von der Klasse 'GenDataAttributeClass' gehalten werden können. Theoretisch ist es ebenso möglich, auf diese Klasse zu verzichten und die Generalisierung, die auf sie zeigt, auf die Klasse 'GenDataAttributeClass' zeigen zu lassen. Dies hätte jedoch zum Nachteil, dass bei der Vererbung viele Attribute mitvererbt, aber nicht gebraucht werden und so Speicherplatz verschwendet würde. Eine Erweiterung des Modells stellen ebenfalls die Report-Control-Klassen sowie die DataSet-Klasse dar. Sie sind zwar nicht direkt im Meta-Modell der Norm angegeben, sind aber ein integraler Bestandteil dieses Standards, da sie einen wichtigen Baustein innerhalb eines Logical Node darstellen. Die Report-Control-Klassen sind für das Versenden von Nachrichten je nach Ereignis zuständig, wobei es möglich ist, mehrere Nachrichten zunächst eine bestimmte Zeit zu puffern (BRCB) und dann zu versenden oder dies sofort beim eintreffenden Ereignis zu tun (URCB). Die DataSet-Klasse ermöglicht den Zugriff auf eine Menge von referenzierten Datenpunkten, dies ist auch über mehrere Logical Nodes hinweg möglich. Referenziert werden Daten und

Objekte über eindeutige Namen bzw. Referenzen.

Ferner wurde auf die Klasse 'GenSubDataAttributeClass' verzichtet. Es wäre möglich, sie als Klasse beizubehalten und Instanzen dieser Klasse als Liste in die 'GenConstructedAttributeClass' einzubetten. Dies würde jedoch eine weitere Kapselung der tatsächlichen Daten bedeuten und damit das Schema unnötig komplexer machen, insbesondere im Bezug auf eine Umsetzung. Aus diesem Grund trägt die 'GenConstructedAttributeClass' die Liste untergeordneter Attribute, die wiederum vom Typ 'GenDataAttributTypeClass' sind. Angewendet wurde hier das Entwurfsmuster 'Kompositum', um eine einheitliche Behandlung von Primitiven (Basetype, CommonACSIType) und Kompositionen (GenConstructedAttributeClass) zu ermöglichen, und um eine einfache Erweiterbarkeit sicherzustellen, da die Klasse 'CommonACSIType' nicht nur eine Klasse ist, sondern eine Menge von Klassen repräsentiert. Der Übersicht halber sind nicht alle möglichen Klassen angegeben. Die Norm IEC 61850 definiert einige Klassen diesen Typs, die jedoch im Kontext dieser Arbeit eine untergeordnete Rolle spielen.

Eine Identifizierung des Typs findet in allen Klassen stets über den Namen oder eine ID statt, da dies ein generisches Modell ist und sich eine tatsächliche Ausprägung, also eine Instanz dieser Klasse, erst durch die Konfiguration über eine entsprechende Datei entsteht. Die Triggeroptionen wurden ebenfalls als eigene Klasse modelliert. Diese Auslagerung ist im Standard nicht definiert, bietet sich als Kapselung jedoch an, um spezielle Funktionen zu definieren und um die einzelnen Optionen zu Kapseln. Diese Triggeroptionen geben an, bei welchem Ereignis ein Report über einen 'Report Control Block' erfolgen soll.

Das in Abbildung 4 dargestellte Modell beschreibt die sehr grundlegende und daher auch generisch gehaltene Struktur des Datenmodells. Erkennbar ist, dass die Klasse 'GenLogicalNodeClass' eine zentrale Rolle einnimmt. Sie hält die verschiedenen 'DataObjects', Instanzen der 'GenDataObjectClass' Klasse, die die Datenpunkte halten. Ein Logical Node stellt somit eine zentrale Einheit im IEC 61850 Modell dar. Jedoch fehlt zum Verbindungsaufbau und Datenaustausch noch eine Datenstruktur, die als Server fungiert. Sie soll Verbindungen aufbauen und abbauen sowie die einzelnen Logical Nodes bzw. Logical Devices kapseln. Abbildung 5 zeigt die Erweiterung des Modells aus Abbildung 4. Diese Erweiterung ist von der Norm IEC 61850 beschrieben, jedoch nicht formal definiert. Die Klasse 'GenLogicalDeviceClass' soll als eine Art Container für die Logical Nodes dienen. Die Klasse 'FileTransferClass' ist für den Austausch von Dateien zuständig. Für die betrachteten Anwendungsfälle ist diese Klasse hauptsächlich für die Übertragung der Konfigurationsdateien verantwortlich. Auch die weitere Logik zur Umsetzung der (Neu-)Konfiguration soll hier implementiert werden. Der Standard definiert für das Informationsmodell keine Strukturren im Metamodell, die die Konfiguration übernehmen. Aus diesem Grund müssen weitere Services bzw. Strukturen entworfen werden, die die Aufgaben der Konfiguration übernehmen. Dies soll standardkonform geschehen (siehe Abschnitt 4.5).

Neben den reinen Daten sind für die Klassen ebenfalls verschiedene Services definiert (siehe Tab. 2). Über sie soll später der Datenaustausch bzw. Informationsaustausch stattfinden. Auch die Übertragung von Informationen ist von der IEC 61850 Norm in Objekten definiert und so existiert zu jedem Service je eine 'Request-Klasse',

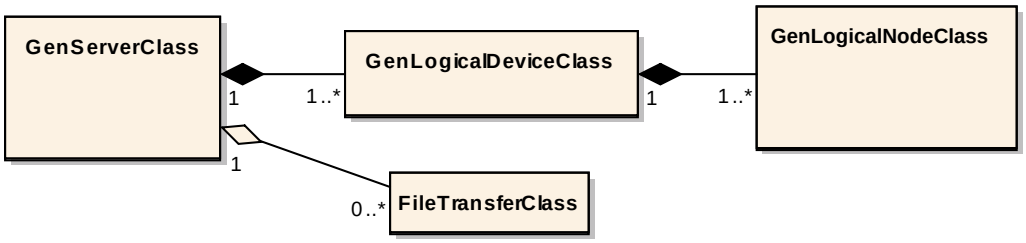

Abbildung 5: Server Meta Modell Erweiterung nach [IEC 61850-7-2]

die die entsprechende Anfrage definiert sowie je eine 'Response-Klasse' für eine positive und negative Rückmeldung. Welche Logical Nodes eine IED hält und auch die von ihr zur Verfügung gestellten Services sollen in einer Konfigurationsdatei beschrieben werden. Über diese sogenannte Selbstbeschreibung der einzelnen Geräte ist den anderen Geräten sowie einer Leitstelle bzw. Kontrollstation stets bekannt, welche Daten abgerufen werden können. Darüber hinaus soll es möglich sein, über eine solche Datei eine neue Konfiguration des Gerätes vorzunehmen (siehe Abschnitt 4.5). Damit eine solche Datei übertragen werden kann, ist von der Norm IEC 61850 ebenfalls ein 'File transfer model' definiert. Neben einer Klasse zur Beschreibung von Dateien sind Services definiert, die den Austausch zwischen Client und Server ermöglichen. Dieses Modell wird ebenfalls innerhalb der Server-Klasse umgesetzt. Der Server koordiniert neben den Verbindungen zu den Clients auch die Ausführung von Services und stellt somit das für den Client sichtbare Interface dar.

Dieses Modell, das aus der Beschreibung der Norm IEC 61850 abgeleitet und entwickelt wurde, bietet die Grundlage dafür, dass die Anforderungen, die in dieser Arbeit an das Batteriesystem gestellt werden, erfüllt werden. Es existieren Klassen für die Datenhaltung und für das Aufnehmen von Sollwerten, für das Versenden von Reports und Klassen für die Konfiguration. Ihnen fehlt jedoch noch die jeweilige Semantik und Logik sowie entsprechende Services zum Austausch der Daten, um die entsprechenden Aufgaben zu erfüllen. Auf der Grundlage dieses Meta-Modells können nun spezielle Instanzen von z.B. Logical Nodes und Common Data Classes erstellt werden. Diese Instanzen werden durch eine Referenz bzw. einen Namen identifiziert und bilden je nach Aufgabengebiet verschiedene Messpunkte, Einstellungen und Befehlsfunktionen ab. Im folgenden sollen diese spezialisierten Instanzen für ein Batteriesystem erarbeitet und erläutert werden.

4.4 IEC 61850 konformes Datenschema für Batteriespeicher

Auf dem zuvor erarbeiteten Meta-Modell sollen folgend jene Bausteine identifiziert werden, die für ein IEC 61850 konformes Informationsmodell für Batteriespeicher notwendig sind. Dazu gehört neben der Auswahl der speziellen Logical Nodes, Common Data Classes und Services auch die Ergänzung jener Strukturen, so dass die Anforderungen erfüllt werden können. Einige Datenklassen sowie Services, die vom Standard definiert werden, sind für diese Anwendung nicht notwendig und werden aus diesem Grund nicht weiter berücksichtigt.

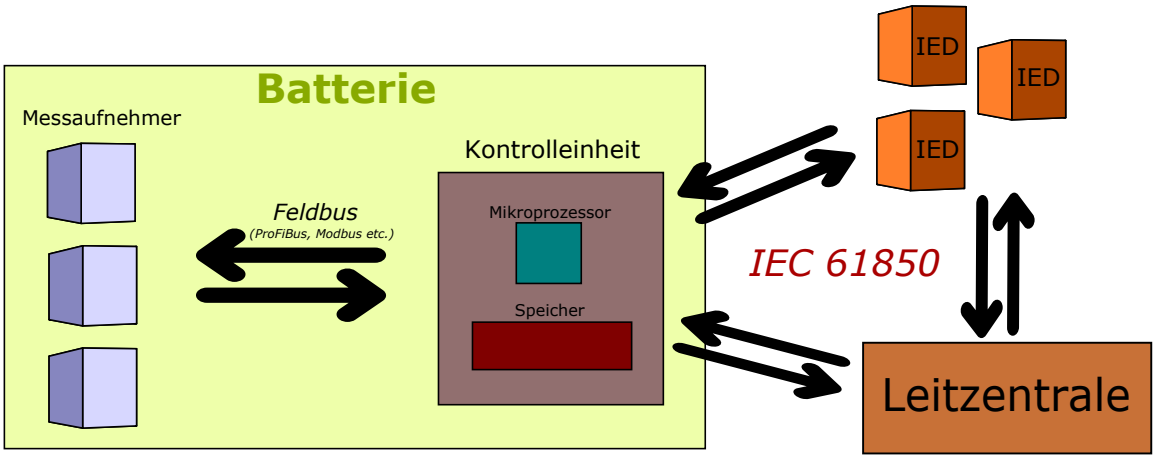

Abbildung 6: Kommunikationsmodell

Abbildung 6 zeigt schematisch das Einsatzgebiet der Norm IEC 61850. Die in dieser Arbeit entwickelten Datenstrukturen und Services bilden primär die Kommunikation zwischen dem Batteriesystem und einer Leitzentrale ab. Wobei die Leitzentrale hauptsächlich für die Steuerung und Überwachung von mehreren IEDs zuständig ist und je nach Umfang des Systems verschiedene Ausprägungen hat. Neben einer reinen Kommunikation zwischen Batterie und Leitzentrale ist auch der Datenaustausch zwischen IEDs und der Batterie möglich, indem die zur Verfügung gestellten Services von anderen IEDs genutzt werden. Neben dem Einsatzgebiet der Norm IEC 61850 ist ebenfalls der Datenaustausch zwischen batterieinternen Messaufnehmern, bespielsweise für Spannung, Wirk- und Blindleistung, abgebildet. Hier können verschiedene Feldbusse zum Einsatz kommen. Der Datenaustausch zwischen Messaufnehmern und der Kontrolleinheit ist jedoch nicht zentraler Bestandteil dieser Arbeit und wird neben kurzen Erläuterungen bei der Implementierung als gegeben angenommen.

4.4.1 Entwicklung des Datenschemas für Batteriespeicher

Die Entwicklung des speziellen Datenschemas für Batteriespeicher ist eng verknüpft mit den Anwendungsfällen, die in dieser Arbeit dargelegt sind. Folgend soll anhand dieser Anwendungsfälle bzw. konkret anhand der Tabelle 9 aus Kapitel 3 eine Auswahl an Logical Nodes getroffen werden, so dass alle Daten des Batteriesystems erfasst werden.

Zunächst werden die beiden Logical Nodes *LLN0* und *LPHD* benötigt, da die Norm IEC 61850 vorschreibt, dass in jedem IED genau jeweils eine Instanz dieser Knoten existieren muss. Der ‘Logical Node Zero’ enthält verschiedene allgemeine Informationen, wie beispielsweise den Zustand des Gerätes. Sehr ähnlich verhält es sich mit dem Logical Node für ‘Physical Device Information’ LPHD, der allgemeine Informationen zum tatsächlichen physikalischen Gerät enthält. In beiden Knoten lassen sich ebenfalls Daten zum Eigentümer hinterlegen. Darüber hinaus ist der Logical Node LLN0 für das Logging zuständig. Er hält Instanzen des ‘Log Control Blocks’, die für Zugriffe auf den Log zuständig sind. Wie der Log lokal abgebildet wird, ist im Standard nicht weiter definiert, da dies je nach Technologie des IED unterschiedlich zu behandelt ist.

Für battriespezifische Daten sind die Logical Nodes 'Battery Systems' *ZBAT* und 'Battery Charger' *ZBTC* zuständig. Der Knoten ZBAT hält allgemeine Informationen zur Batterie. Dazu gehört neben dem Typ der Batterie auch die Kapazität sowie Informationen zum Entladevorgang (Einspeisung). Ferner können über diesen Logical Node Tests ausgeführt und das System an- bzw. ausgeschaltet werden. Weiterhin kann über einen Fahrplan (Scheduling) das Einspeiseprofil kontrolliert werden. Der 'Battery Charger' Knoten hält Daten zum Ladevorgang wie beispielsweise Spannung und Strom und kontrolliert ihn über einen Schedulingplan, der hinterlegt werden kann. Beide Logical Nodes haben außerdem die Möglichkeit, verschiedene Messwerte aufzunehmen, um jedoch alle relevanten Messdaten abzubilden sind weitere Knoten notwendig.

Die Abbildung der verschiedenen Daten eines Dreiphasensystems ist Aufgabe des Logical Node 'Measurement' *MMXU*. Dieser Knoten hält die Daten für die Spannung, Strom, Phase, Frequenz, Leistung etc. und ist damit wichtig für die Überwachung des Batteriesystems. Die Messdaten für ein Gleichstromsystem sollen vom Logical Node 'DC Measurement' *MMDC* gespeichert werden. Neben diesen Messwerten sollen außerdem Daten gespeichert werden, die für eine Abrechnung zwischen z.B. Netz- und Batteriebetreiber wichtig sind. Dies ist Aufgabe des Logical Nodes 'Metering' *MMTR*. Er zeichnet auf, wie viel Enegie tatsächlich zur Verfügung gestellt bzw. geladen wurde. Daneben kann es sinnvoll sein, generelle Informationen zu einem Vertragsverhältnis festzuhalen, wie beispielsweise das Limit der Einspeisung oder der Aufnahme an elektrischer Energie sowie die festgelegte Wirk- oder Blindleistung. Für diese Daten ist der Logical Node 'Economic dispatch parameters' *DCCT* vorgesehen.

Ein wichtiger Bestandteil eines Batteriesystems stellt der Wechselrichter dar, denn das Laden der Batterie erfolgt mit Gleichstrom wobei zur Übertragung des elektrischen Stroms zumeist Wechselstrom genutzt wird. Die Kontrolle über dieses Gerät soll mit dem Logical Node 'Inverter' *ZINV* erfolgen. Mit diesem Knoten lassen sich Sollwerte für die Wirk- und Blindleistung sowie für die Frequenz vergeben. Desweiteren lassen sich die drei Phasen konfigurieren und es können allgemeine Informationen zum Wechselrichter hinterlegt werden.

In Tabelle 16 in Anhang B.1 sind alle ausgewählten Logical Nodes aufgeführt. In der Tabelle sind außerdem die Common Data Classes sowie Datenattribute dargestellt, die die Batterieinformationen halten. Außerdem sind Common Data Classes aufgeführt, die keine Angabe einer Information haben. Dies sind Klassen, die die Norm IEC 61850 als 'mandatory' angibt, also obligatorisch sind und somit im entsprechenden Logical Node existieren müssen. Bis auf einen Datenpunkt können alle Batterieinformationen mit Semantik entsprechend abgebildet werden. Das fehlende Datum ist der State-of-Charge, der aktuelle Ladestand der Batterie. Die Norm definiert kein 'Data Object' in einem Logical Node, dass für diesen Datenpunkt vorgesehen ist. Dieses Datum stellt jedoch eine essentielle Information für den Betrieb eines Batteriesystems dar und muss aus diesem Grund hinzugefügt werden. Da der State-of-Charge ein Datum darstellt, welches zum Ladevorgang der Batterie gehört wird es dem Logical Node ZBTC hinzugefügt.

Der Ladestand wird hier als Common Data Class 'Measured Value' MV modelliert. Da der Ladestand ein Messwert ist, bieten sich drei im Standard definierte Common Data Classes an. Zum einen die Klasse 'Sampled Value' SAV, sie ist für die Speicherung von Echtzeitwerten zuständig. Die Klasse 'Complex Measured Value' speichert

komplexere Messwerte wie beispielsweise Werte für die verschiedenen Phasen eines Drei-Phasen-Systems. Die Klasse 'Measured Value' MV ist für allgemeine Messwerte vorgesehen und daher nicht weiter spezifiziert. Somit lässt sich der Ladestand am besten durch die Common Data Class MV darstellen. Tabelle 13 bildet die Datenattribute jener Klasse ab. Es sind nur die obligatorischen bzw. notwendigen Attribute aufgelistet. Das Attribut 'mag' hält den tatsächlichen Messwert des Ladestands und 'units' die entsprechende Einheit. Zusätzlich kann noch die Qualität sowie der Zeitpunkt der Messung festgehalten werden.

Tabelle 13: Common Data Class 'MV' für den Ladestand

Data Attribut Name	Typ	FC	Trigger Option
mag	AnalogueValue	MX	dchg, dupd
q	Quality	MX	dchg
t	TimeStamp	MX	—
units	Unit	CF	dchg

Für alle relevanten Daten, die in der Analyse identifiziert wurden (Tabelle 9), sind nun die entsprechenden Datenstrukturen, die auch die Semantik berücksichtigen, ausgewählt sowie erweitert worden. Um die weiteren Anforderungen zu erfüllen sollen folgend die Strukturen erläutert werden, die den Austausch der Daten ermöglichen. Daran schließt sich die tatsächliche Auswahl an Services, die die Kommunikation umsetzen, an.

Der Hauptaspekt ist die Übertragung von (Mess-)Werten. Dazu soll die Klasse 'DataSet' genutzt werden. Sie hält eine Liste an Referenzen der verschiedenen Datenattribute (GenDataAttributeClass) und ermöglicht so das Setzen bzw. Abrufen jener Datenwerte. Das 'DataSet' ist flexibel einsetzbar und ist ein Objekt des entsprechenden Logical Nodes. Es kann vom Client erstellt werden und bietet die Möglichkeit der individuellen Auswahl an Attributen. Das Verändern von Datenattributen wäre auch über die Klasse 'GenDataObjectClass' mit den Services 'GetDataValues' und 'SetDataValues' möglich. Diese Services ermöglichen jedoch nur das Editieren von einem Attribut bzw. aller Attribute auf einmal. Das Editieren aller Attribute ist nicht zielführend, da es in der späteren Umsetzung nicht vorkommt, dass alle Daten eines 'Data Objects' gleichzeitig neu gesetzt werden müssen. Weiterhin ist es nicht möglich, mit diesen Services eine Teilmenge an Attributen zu adressieren. Ein Data Set mit entsprechenden Services ist dieser Möglichkeit also vorzuziehen. Ein Nachteil des 'DataSets' ist jedoch, dass auch hier vom Standard vorgegeben wird, dass die Services in einer Anfrage stets alle referenzierten Daten abrufen bzw. setzen müssen. Sinnvoll wäre es, die gewünschten Attribute bei einem Aufruf des Services mit angeben zu können, so muss je nach Anwendung ein individuelles Data Set angelegt werden. Ein weiterer Vorteil des Data Sets besteht in der Möglichkeit diese ebenfalls über eine Konfigurationsdatei definieren zu können. Somit ist es möglich 'DataSets' direkt bei der Konfiguration eines Gerätes zu erstellen. Des Weiteren können sie von 'Report Control Blocks' referenziert werden. So können die referenzierten Attribute überwacht werden und es kann ein Report bei entprechenden Ereignissen erfolgen.

Die Anforderung der (Mess-)Wertübertragung ist den Anforderungen der Vorgabe von Sollwerten und dem Versenden bzw. Empfangen von Fahrplänen recht ähnlich. Die Vorgabe von Sollwerten und insbesondere das Senden von Befehlen, also das Steuern des Systems ist von der Norm IEC 61850 durch das 'Control Model' definiert. Möglich ist es hingegen auch, Sollwerte und Befehle über die Nutzung der zuvor behandelten Services der 'DataSets' zu realisieren. Das 'Control Model' ist sehr umfangreich und bietet insbesondere die Möglichkeit, die Befehlsausführung abzusichern. Die Norm realisiert dies über das 'Select-Before-Operate'-Prinzip sowie verschiedener Zustandsdiagramme. Bevor der Befehl ausgeführt wird, wird zunächst vom Client eine Select-Anfrage gestellt und je nach Zustandsdiagramm (normale/höhere Sicherheit) wird dann entschieden, ob der Client berechtigt ist und ob die Anfrage zulässig ist. Bekommt der Client darauf vom Server eine positive Nachricht, so kann dieser den Operate-Service mit dem entsprechenden Befehl für eine gegebene Zeit ausführen. Je nach Ausführung wird darauf vom Server mit einer positiven oder negativen Nachricht geantwortet. Dieses 'Control Model' ist laut Norm nur auf die 'Controllable Common Data Classes' (siehe Anhang) anwendbar. Eine solche Common Data Class findet sich bei den ausgewählten Logical Nodes nur unter ZBTC: SPC (Controllable Single Point). Sie ist für den Batterietest und das Ein- und Ausschalten der Batterie zuständig. Hier ist es sinnvoll, das 'Control Model' anzuwenden, da es sich hier um kritische Befehle handelt, die zumeist auch einer Autorisierung bedürfen. Aufgrund der Komplexität soll jedoch eine tiefergehende Betrachtung dieses Modells in dieser Arbeit nicht erfolgen. Die Vorgaben für die Wirkleistung, Blindleistung und Frequenz sollen über die 'DataSet' Klasse und die zuvor genannten Services vorgenommen werden. Die Kontrolle über das Einspeisen bzw. Speichern von elektrischem Strom wird über das Vorzeichen der Leistungen realisiert. Sinnvoll kann die Trennung in verschiedene 'Data Sets' sein, damit die Leistungen und die Frequenz einzeln editierbar sind.

Für das Ablegen von Fahrplänen bzw. Schedulings gibt die Norm die Common Data Class CSG (Curve Shape Setting) (siehe Tabelle 17) vor. Diese kann zwei Arrays mit Datenwerten aufnehmen, wobei die eine Menge die Ordinatenwerte speichert und die andere Menge die Abszissenwerte. Je nach Anwendungsgebiet werden die Einheiten der Achsen außerdem gespeichert. So lassen sich Funktionen durch diskrete Werte nachbilden oder in diesem Fall Schedulings für die Einspeisung bzw. das Laden der Batterie abbilden. So wird als Einheit für die Ordinate die Einspeise- bzw. Ladeleistung angegeben und als Einheit für die Abszisse die Zeit gewählt. Nun kann das Batteriesystem lokal für die entsprechenden Zeiten die Leistung anpassen. Für die Übertragung dieser Fahrpläne eignen sich ebenfalls die Services der 'DataSet'-Klasse. Auch hier ist die Zusammenfassung von Ordinaten- und Abszissenwerten sowie den jeweiligen Einheiten zu einem 'Data Set' sinnvoll, da diese in der Regel gleichzeitig gesetzt werden. Weitere Möglichkeiten, ein Scheduling abzubilden, wie z.B. über Textdateien, sind nicht standardkonform möglich und erzielen ebenso keine Verbesserung gegenüber der beschriebenen Methode.

Zu den weiteren Anforderungen an das System zählt das Senden von Reports, also die Benachrichtigung der Kontrollstruktur z. B. der Leitstelle. Realisiert werden könnte dies durch einen einfachen Service, der je nach Ereignis, Datenattribut und entsprechender Einstellung eine Nachricht versendet, die Aufschluss darüber gibt, wie beispielsweise sich der Wert geändert hat oder ob ein Update des Attributes erfolgt ist. Dies stellt eine

Minimallösung dar und ist ausschließlich als Service nicht standardkonform umsetzbar. Ein Service muss immer im Kontext einer Klasse eingebettet sein. Außerdem ist nicht definiert, an welche Adresse dieser Report gesendet werden soll. Denkbar wäre hier eine Broadcast-Nachricht. Dies hätte jedoch eine schlechte Ausnutzung der Bandbreite zur Folge, da es nur einen Empfänger gibt und Reports relativ hochfrequentiert versendet werden sollen. Aus diesem Grund soll eine Klasse als Kontrollinstanz das Versenden von Reports organisieren und individuell für ein Datenattribut entscheiden, wann ein Report zu senden ist und an welche Adresse dies erfolgen soll. Der Standard sieht für diesen Zweck die Klassen Buffered- und Unbuffered-Control-Block vor. Diese speichern dazu unter anderem eine Referenz zu einem Data Set. Abhängig vom jeweiligen Ereignis und den Triggeroptionen, der vom Data Set referenzierten Datenattribute, wird dann vom Report-Control-Block eine Nachricht versendet. Nach Bedarf kann ein Client einen solchen Control-Block mit entsprechendem Data Set erstellen und so die Reports für die referenzierten Datenattribute empfangen. Daneben können diese Kontrollblöcke auch über die Konfigurationsdatei direkt instanziiert werden. Das ungepufferte Versenden von Reports ist für die in dieser Arbeit betrachteten Anwendungsfälle hinreichend, da kein Anwendungsfall die Anforderung an eine Pufferung von Reports stellt. Für das Puffern bzw. Speichern von Ereignissen soll das Logging zuständig sein.

Für das Logging ist von der Norm IEC 61850 eine ‘LOG’-Klasse sowie ein entsprechender Kontrollblock (Log-Control-Block) definiert. Die Norm sieht vor, dass das Logging unabhängig von der Kommunikation ist und somit auch bei Verbindungsausfällen Ereignisse festgehalten werden. Das Logging ist somit komplett asynchron zu jeglichem Senden und Empfangen von Nachtrichten. Darüber hinaus sind die Log - Daten persistent zu speichern, so dass diese auch nach Stromausfällen noch vorhanden sind. Die ‘LOG’-Klasse der Norm kapselt die tatsächlichen Ereignisdaten und speichert außerdem den Zeitstempel und die Sequenznummer des letzten und ersten Eintrages. Jeder Eintrag hat eine ID, einen Zeitstempel, eine Referenz zum aufgenommenen Attribut sowie dessen Wert. Gespeichert werden kann außerem der Grund des Loggings. Wie genau diese Daten persistent gespeichert werden, ist von der Norm nicht definiert. Vorgeschlagen wird, aufgrund von möglichem Speichermangel bei eingebetteten Systemen, einen Ringpuffer zu verwenden und so bei vollem Speicher alte Einträge zu überschreiben. In diesem Zusammenhang ist die einfache Speicherung in einer Textdatei (z.B. CSV) denkbar. Ist ein größerer Speicher verfügbar ist, auch der Einsatz eines Datenbanksystems möglich. Interessant könnte auch der Einsatz eines Zentralen Speicherblocks für mehrere IEDs sein. Damit wäre das Logging nicht mehr unabhängig vom Senden und Empfangen von Nachrichten, ein ‘Trade-off’ aus temporärer Speicherung von Ereignissen lokal in Textdateien, die dann in Intervallen im zentralen Speicher gespeichert werden, könnte hier jedoch eine Problemlösung darstellen. Die ‘LOG’-Klasse, die die Ereignisdaten kapselt, wird wiederum von einem ‘Log-Control-Block’ referenziert und überwacht. Über ihn kann das Logging ein- bzw. ausgeschaltet werden. Zudem referenziert er das zu protokollierende Data Set und legt fest für welche Triggeroptionen ein Log angelegt werden soll. Es ist weiterhin möglich, dass mehrere Kontrollblöcke auf den selben Log schreiben. Sowohl LCB als auch LOG sind über einen Namen bzw. über eine eindeutige Referenz identifizierbar und werden ausschließlich vom Logical Node LLN0 gehalten. Dies ist auch der Grund, warum jene Klassen nicht im Meta-Modell enthalten sind, da sie diesbezüglich nicht generisch sind und nur zu einer Instanz der Logical

Nodes gehören. Das entsprechende UML Diagramm für das Logging ist in Anhang A.2 zu finden.

Das Konzept für die Konfiguration von IEDs über entsprechende Dateien und Services ist Inhalt des Abschnitts 4.5 und somit getrennt von der restlichen Konzeptionierung, da es im Hinblick auf die Standardkonformität eine gesonderte Rolle einnimmt. An die Konzeption des Datenschemas anhand der Anforderungen, die aus den Anwendugsfälle abgeleitet wurden, soll sich im Folgenden die Auswahl an entsprechenden Services anschließen.

4.4.2 Auswahl erforderlicher Services

Einen essentiellen Teil des 'Abstract Communication Service Interface' stellen die Services dar. Sie bilden die Schnittstelle zwischen den lokalen Daten und dem tatsächlichen Informationsaustausch. Sie erfüllen damit die nichtfunktionelle Anforderung an die Kapselung der Software, indem sie durch eine klar definierte Struktur es ermöglichen, den Austausch der Daten auf verschiedene Arten umzusetzen. Dazu bedarf es ausschließlich des Mappings des ACSI auf ein entsprechendes Kommunikatinsprotokoll (siehe Kapitel 4.6).

Zum bereits entwickelten Datenmodell sollen nun folgend die entsprechenden Services ausgewählt werden, um die Systemanwendungsfälle und die damit verbunden Anforderungen zu erfüllen. Zunächst wird der generelle Aufbau eines Services erläutert und anschließend die speziellen Services der einzelnen Datenstrukturen ausgewählt. Jeder Service gliedert sich in eine Anfrage (Request), eine positive und eine negative Rückmeldung (Response). Die Anfrage enthält alle relevanten Informationen, um eine gewünschte Aktion auszuführen (siehe Tabelle 15), wie z. B. die Referenz eines Data Sets zum Abrufen der referenzierten Attribute. Je nach Erfolg der Anfrage wird dem Client dann eine positive bzw. negative Rückmeldung zurückgesendet. In einer positiven Rückmeldung stehen dann beispielsweise die angefragten Attributwerte. In einer negativen Antwort soll der Grund für ein Fehlschlagen enthalten sein.

Das Übertragen von (Mess-)Werten, Befehlen, Sollwerten und Fahrplänen soll, wie bereits modelliert, über die 'DataSet'-Klasse erfolgen. Für das Abrufen und Speichern von Daten sind die Services 'GetDataSetValues' und 'SetDataSetValues' ausreichend. Für das Versenden von Reports bedarf es ausschließlich des Services 'Report', der je nach referenziertem Data Set und Triggeroptionen einen Report an den entsprechenden Client versendet. Für das Abfragen des Logs soll über die 'LOG'-Klasse der Service 'QueryLogByTime' zur Verfügung stehen. 'QueryLogByTime' soll die Abfrage über einen Zeitraum anhand der Zeitstempel ermöglichen. Die Einstellungen für den Log-Control-Block, die Data Sets sowie den Report-Control-Block sollen über die Konfigurationsdatei möglich sein.

Für einen erfolgreichen Einsatz der Norm IEC 61850 sind neben den Services, die die Systemanwendungsfälle sowie Anforderungen erfüllen, weitere Schnittstellen notwendig. Zum einen werden Services für den Aufbau und den Abbau einer Verbindung zur Kommunikation benötigt. Die Services 'Associate' und 'Release' sind Teil der Serverdatenstruktur und implementieren eben diese Funktionen. Informationen zum Aufbau der Datenstrukturen können über die Konfigurationsdatei eingesehen werden. So

kann auf die “Directory”-Services verzichtet werden, die Aufschluss darüber geben, aus welchen Logical Devices z. B. ein Server besteht oder welche ‘DataObjects’ ein Logical Node besitzt etc. Die Tabelle 15 gibt eine Übersicht über die benötigten Services, entsprechenden Nachrichtentypen sowie über die Informationen, die zu übertragen sind.

Tabelle 14: Übersicht über Services

Service Name	**Nachrichtentyp**	**Informationen**
GetDataSetValues	Request	DataSetReference
	Response+	DataSetReference, Attributwerte
	Response−	Service Error
SetDataSetValues	Request	DataSetReference, Attributwerte
	Response+	Service Success
	Response−	Service Error
Report	Report	Reportformat
QueryLogByTime	Request	LogReference, Startzeit, Endzeit
	Response+	Log-Einträge
	Response−	Service Error
Associate	Request	ServerAccessPoint, AuthenticationParameter
	Response+	Verbindung erfolgreich, AssociationID
	Response−	Service Error
Release	Request	AssociationID
	Response+	Abbau erfolgreich, AssociationID
	Response−	Service Error

4.5 Instanziierung aus Konfigurationsdateien

Als eine besondere Anforderung an das System ist die (Neu-)Konfiguration gegeben. Es soll demzufolge möglich sein, die Datenrepräsentation sowie die zur Verfügung gestellten Services des IED je nach Anwendung ändern zu können. Die Normenreihe IEC 61850 definiert für diesen Zweck eine Konfigurationssprache auf XML-Basis. Diese ‘Substation Configuration Language’ (SCL) ist detailliert im Normenteil [IEC 61850-6] beschrieben. Für jede Datenstruktur, die das Meta-Modell vorgibt, ist ein entsprechendes SCL-Tag definiert (siehe Anhang C). Der verschachtelte Aufbau einer XML-Datei ist hervorragend für die Abbildung der hier beschriebenen Datenmodelle geeignet, da diese ebenfalls diese Struktur aufweisen: Ein Server enthält mehrere Logical Devices, diese enthalten wiederum mehrere Logical Nodes etc. Neben der Beschreibungssprache definiert der Standard ebenfalls ein ‘File Transfer Model’, welches den Dateiaustausch zwischen Client und Server ermöglicht und Teil der Serverdatenstruktur ist. Somit ist gewährleistet, dass eine Konfiguration auch ferngesteuert z. B. über das Internet möglich ist.

Aus diesen beiden Komponenten lässt sich jedoch nicht sofort eine Übertragung einer Konfigurationsdatei und die anschließende Konfiguration des Systems aufbauen. Der Standard hat für die Konfiguration eines IED keine eigenständigen Services definiert und mit den Services des 'File Transfer Models' lässt sich die Datei nur auf den Server übertragen. Denkbar wäre die Erweiterung des 'SetFile'-Services, so dass es möglich ist eine Art Flag oder Wert zu setzen der indiziert, dass nach Übertragung dieser Datei ebenfalls eine Neukonfiguration des Systems anhand dieser Datei zu erfolgen ist. Für eine semantische Trennung zwischen der Übertragung allgemeiner Dateien und der Übertragung von Konfigurationsdateien könnte ein eigener Service entwickelt werden, der nur SCL-Dateien akzeptiert und das System dazu veranlasst, eine Neukonfiguration durchzuführen. Beide Erweiterungen hätten jedoch zur Folge, dass die Umsetzung nicht mehr standardkonform ist, da der Standard einen Entwurf von eigenen Services bzw. die Veränderung von bestehenden Services nicht vorsieht. Um dennoch eine Standardkonformität zu gewährleisten, soll eine Neukonfiguration stets dann vorgenommen werden, wenn ein Client über den 'SetFile'-Service eine Konfigurationsdatei auf das Batteriesystem überträgt. Dies soll serverseitig durch eine Prüfung der Dateiendung '.icd' geschehen. Es schließt sich an jede Übertragung einer solchen Datei stets eine Konfiguration des Systems an, dies kann akzeptiert werden, da eine Übertragung einer solchen Datei aus anderen Gründen nicht zweckmäßig ist. Für die Übermittelung der Datei soll der 'SetFile'-Service verwendet werden. Es ist für keinen Anwendungsfall von Nöten, dass eine Datei auch wieder vom Batteriesystem heruntergeladen werden muss. Aus diesem Grund kann auf einen 'GetFile'-Service verzichtet werden. Es ist jedoch denkbar, dass aufgrund von mangelndem Speicherplatz Dateien gelöscht werden müssen. Dies soll über den 'DeleteFile'-Service geschehen. Durch die Angabe des Dateinames kann die zu löschende Datei identifiziert werden.

Tabelle 15: File-Transfer Services

Service Name	**Nachrichtentyp**	**Informationen**
SetFile	Request	Dateiname, Datei
	Response+	Speicherung/Konfiguration erfolgreich
	Response−	Service Error
DeleteFile	Request	Dateiname
	Response+	Löschung erfolgreich
	Response−	Service Error

Nach dem erfolgreichen Empfang einer Konfigurationsdatei soll diese ein Parsing durchlaufen, um die einzelnen Datenstrukturen zu identifizieren. Es soll hier stets davon ausgegangen werden, dass diese Datei richtig strukturiert und aufgebaut ist. An das Parsing schließt sich die Instanziierung der einzelnen Datenstrukturen an. Je nach angegebener Struktur wird dann aus dem Meta-Modell das respektive Objekt instanziert und die entsprechenden Attribute und Werte gesetzt. Ist die Neukonfiguration erfolgreich verlaufen, ist dies dem Client in einer positiven Response-Nachricht mitzuteilen. Kam es zu Fehlern, ist ein Service Error zu versenden.

Für die Anwendungsfälle und speziell für die Anforderungen an das Batteriesystem

konnten nun jene Teile konzeptioniert und erarbeitet werden, um die erforderlichen Daten zu halten und sie über abstrakte Services auszutauschen. Das batteriespezifische 'Abstract Communication Service Interface' ist somit definiert sowie in sich geschlossen und bietet durch diese Kapselung die Möglichkeit, den tatsächlichen Austausch von Daten über verschiedene Methoden variabel zu gestalten.

4.6 Mapping auf MMS

Dieses Kapitel beschäftigt sich mit dem Mapping der zuvor definierten Datenstrukturen und Services auf ein Protokoll, dass für die Übertragung der Daten über ein Netzwerk zuständig ist. Folgend soll also diskutiert werden, welche Protokolle und Transportstrukturen sich für die Übermittelung der Daten am besten eignen.

Grundsätzlich ist ein Mapping auf jedliche Transportprotokolle denkbar. Beispielsweise könnten die Objekte, Attribute und Services direkt auf das 'Transmission Control Protokoll' (TCP) abgebildet werden. Hierzu wäre es nötig, eine String- bzw. Textrepräsentation der Objekte zu definieren. Diese Daten könnten dann als Nutzlast des TCP-Pakets versendet werden, wobei auch der Empfänger die Textrepräsentation kennt und so die Daten richtig interpretieren kann. Eine Kapselung zwischen Applikations- und Transportschicht wäre so allerdings nicht gegeben. Es könnten jedoch Dienste für die jeweiligen Services wie Verbindungsaufbau- und abbau, Reports, Get- und Setter-Services etc. entwickelt und so eine entsprechende Kapselung erreicht werden. Die Entwicklung eines solchen Systems erfüllt die Aufgaben, für die die Norm IEC 61850 den Standard 'Manufacturing Messaging Specification' (MMS) ausgewählt hat. Dieser dient dem objektorientierten Austausch von Daten, definiert Objekte und Dienste zur Übertragung jener Daten und ist losgelöst von der Transportschicht sowie der verwendeten Netzphysik.

In Abschnitt 4.1 wurden bereits Lösungsansätze für die Übertragung der Batteriedaten diskutiert. Die Verwendung eines dieser Protokolle ist theoretisch möglich, jedoch insbesondere aufgrund der fehlenden Logik für ein Mapping, also die entsprechenden Datenrepräsentationen, sowie im Hinblick auf die Standardkonformität der Umsetzung nicht zielführend.

Interessant ist der Datenaustausch über (Representational State Transfer) REST und damit über das Hypertext Transfer Protocol (HTTP). Die meisten Services können über einen Uniform Resource Locator (URL) und den damit verbundenen HTTP-Request bzw. Response abgebildet werden [PPH+10]. Jedoch wird in [PPH+10] gezeigt, dass die Umsetzung von Reports nicht einfach möglich ist, da REST explizit verbindungslos ist. Eine Umsetzung wird ebenfalls nicht weiter beschrieben. Der Report Service ist hingegen Teil des zuvor konzeptionieren Informationsmodells.

Die Norm IEC 61850 definiert in [IEC 61850-8-1] ein Mapping des ACSI auf MMS. Das Mapping des in dieser Arbeit definierten Informationsmodells soll folgend erläutert werden. Es werden ausschließlich die Services und Datenstrukturen beschrieben, die für dieses Konzept notwendig sind. Des Weiteren soll die Kommunikation über MMS auf dem TCP sowie auf Ethernet aufbauen und damit die nichtfunktionale Anforderung der Routbarkeit der Informationspakete erfüllen. Dieser Aufbau ist ebenfalls durch [IEC

61850-8-1] standardisiert. Das Mapping der Datenstrukturen und Services des ACSI ist in der Tabelle 18 und der Tabelle 19 in Anhang B abgebildet.

Der ACSI Server, der die Schnittstelle zwischen den IEDs bzw. zwischen dem Batteriesystem und der Leitwarte darstellt, wird durch das vom MMS definierte Virtual Manufacturing Device (VMD) abgebildet. Das VMD baut über die 'Initiate Services' Verbindungen (Associations) auf bzw. ab. Von einem Server können mehrere Verbindungen gleichzeitig verwaltet werden. Je nach Erfolg der Verbindung wird dem Kommunikationspartner eine positive Nachricht mit entsprechender 'AssociateID' zurückgeschickt oder ein entsprechender 'ServiceError' versendet. Für das Versenden von Reports ist der 'MMS Information Report' vorgesehen. Die jeweiligen Daten eines Reports werden auf die entsprechenden MMS Datenstrukturen übersetzt (siehe Tabelle 18). Die Services zum Speichern bzw. Löschen von Dateien werden von dem 'FileSet' bzw. 'FileDelete' Request umgesetzt. Der Erfolg der Ausführung wird jeweils durch das Versenden des entsprechenden 'Response'-Nachricht gekennzeichnet. Ist ein Fehler aufgetreten, wird ein 'ServiceError' übermittelt. Der Service 'QueryLogByTime' wird durch den 'MMS ReadJournal-Request' abgebildet. Die Referenz auf den Log wird in einen 'journalName' übersetzt und die Anfangs- bzw. Endzeit in einer 'TimeOfDay' Datenstruktur gespeichert. Konnte die Anfrage fehlerlos bearbeitet werden, wird ein 'MMS ReadJournal-Response' mit den angeforderten Logeinträgen zurückgesendet. Das Setzen bzw. Abfragen von Datenwerten wird über die 'Read'- und 'Write-Request' Services umgesetzt. Bei der Abfrage wird stets eine Liste an Attributwerten zurückgesendet. Es ist jeweils angegeben, ob der Zugriff auf das entsprechende Attribut erfolgreich war oder nicht. Beim Setzen von Attributfeldern ist die Liste an Attributwerten in ein ListOfData zu übersetzen. Die Tabelle 19 zeigt das zugehörige Mapping der verschiedenen ACSI Datentypen auf die MMS Datenstrukturen.

Nachdem nun ebenfalls das Mapping definert ist, sind alle notwendigen Teile des Systems erarbeitet worden, um eine Kommunikation für Batteriespeicher zu realisieren.

4.7 Zusammenfassung

In diesem Konzeptkapitel konnte über die Analyse und Bewertung von möglichen Lösungsansätzen und der Entwicklung eines Informationsmodells eine Kommunikation für Batteriespeicher konzeptioniert werden. Anhand der Norm IEC 61850 wurde ein Datenschema und die entsprechenden Services erarbeitet.

Zunächst konnte über einen Vergleich verschiedener Protokolle die Norm IEC 61850 als am geeignetesten herausgestellt werden. Die zuvor aufgestellten Anforderungen an die Kommunikation mit einem Batteriesystem konnte jene Norm am besten erfüllen. Anschließend wurde aus dem im Standard definierten Modell ein UML-Klassendiagramm entwickelt, welches ein Meta Modell für die spätere Objektinstanziierung darstellt. Des Weiteren wurde anhand der Norm das Server Modell mit zugehörigem Dateitransfer Modell konzeptioniert. An die Definition des allgemeinen Meta Modell schloss sich die Konstruktion eines konkreten Datenschemas für Batteriespeicher an. Hier konnten notwendige Bausteine wie Logical Nodes und Common Data Classes anhand der zu kommunizierden Batterieinformationen identifiziert sowie fehlende Teile ergänzt werden.

Daran anschließend wurden die abstrakten, von der Norm standardisierten, Services ausgewählt, die die Schnittstelle zwischen Batteriedaten und dem Kommunikationsstack darstellen. Als eine besondere Anforderung an das System ist die Konfiguration des Informationsschemas gegeben. Hierzu wurden verschiedene Ansätze diskutiert und es konnte eine standardkonforme Lösung über das Dateitransfer Modell erarbeitet werden. Um eine tatsächliche Kommunikation über ein Netzwerk zu ermöglichen, wurde das Mapping des Informationsmodells auf das MMS-Protokoll beschrieben.

Folgend soll die protypische Umsetzung eines Anwendugsfalls anhand des nun erarbeiteten Konzepts beschrieben werden. Diese Umsetzung stellt die Grundlage für die Evaluierung jenes Konzepts dar.

5 Implementierung

Dieses Kapitel soll die prototypische Umsetzung des Anwendungsfalls 'Einspeiseprognose einhalten' erläutern. Zunächst werden die Akteure skizziert sowie Randbedingungen identifiziert. Daran schließt sich die Erläuterung der Programmierung sowie Implementierungsdetails an.

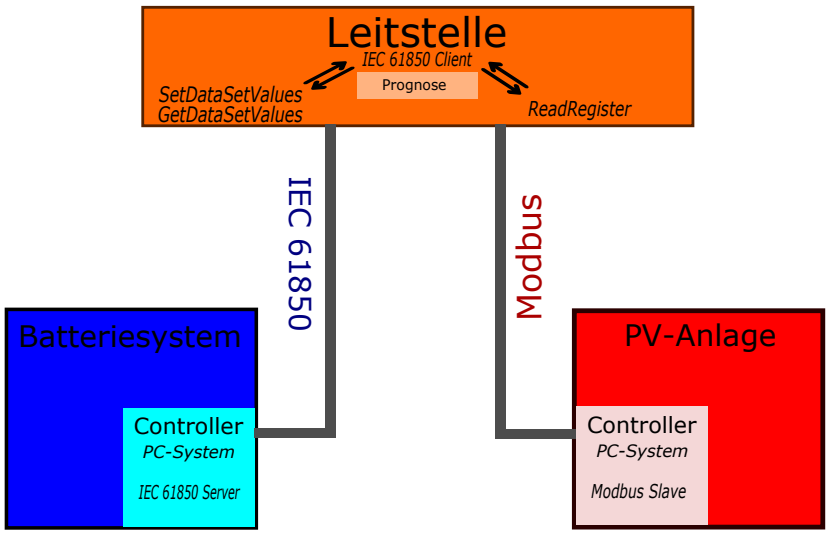

Abbildung 7: Umsetzung des Anwendungsfalls 'Einspeiseprognose einhalten'

Implementiert wurde der Anwendungsfall 'Einspeiseprognose einhalten'. Jenes Szenario ist durch Abbildung 7 verdeutlicht. Je nach Leistung der PV-Anlage sowie vorgegebener Einspeiseprognose, wird von der Leitstelle durch das Senden des 'SetDataSetValues'-Services das Ändern der Einspeise- bzw. Ladeleistung an der Batterie veranlasst. Des Weiteren wird durch den 'GetDataSetValues'-Service der aktuelle Ladestand sowie der aktuelle Ladestrom überwacht. So kann durch das intelligente Steuern der Leitstelle die Batterie genutzt werden, um eine vorgegebene Einspeiseprognose einzuhalten. Ferner sind die IEC 61850 Datenstrukturen, die das Batteriesystem bereitstellt, vorher durch eine Konfigurationsdatei erstellt worden.

5.1 Randbedingungen

Als Randbedingung lässt sich zunächst der Datenaustausch zwischen Leitstelle (PC-System) und PV-Anlage nennen. Die PV-Anlage wird durch einen 'Modbus-Slave[5]' auf einem anderen PC-System simuliert. Ein 'Modbus-Slave' stellt einen Client dar, der Daten über abrufbare Register zur Verfügung stellt. Über eine Modbus TCP Verbindung fragt die Leitstelle in regelmäßigen Zeitabständen die Stromerzeugung (Leistung) der Anlage ab (ReadRegister) und vergleicht sie mit den Werten der Prognose, die lokal auf dem System gespeichert sind. Als weitere Randbedingung, die vom Anwendungsfall nicht gefordert wird, jedoch essentieller Bestandteil des Konzepts ist, ist die Instanziierung der benötigten Datenstrukturen über eine 'SCL'-Konfigurationsdatei. Jene Datei

[5] `http://www.modbustools.com/modbus_slave.html`

wird nicht durch einen 'SetFile'-Service übertragen, sondern ist bereits lokal im Speicher des Batteriesystems vorhanden. Die Instanziierung der Datenstrukturen findet automatisch beim Start des Batteriesystems statt.

Da das in dieser Arbeit entwickelte Konzept für den Einsatz in der Energiewirtschaft entwickelt wurde und dort der Programmcode zumeist in eigebetteten Systemen lauffähig sein muss, wurde sich für die Programmiersprache C++ entschieden. Sie bietet sowohl die Möglichkeit eines hohen Abstraktionsniveaus, als auch die Möglichkeit der ressourcenschonenden und maschinennahen Programmierung.

5.2 Umsetzung

Die Umsetzung des Anwendungsfalls 'Einspeiseprognose einhalten' anhand des Konzepts lässt sich in vier Teile unterteilen.

- Meta Modell
- Instanziierung
- Mapping
- Leitstelle

Begonnen wurde bei der Umsetzung des Konzepts mit der Abbildung der im Meta Modell beschriebenen Klassen in entsprechende C++ Klassen. Dazu wurde zunächst aus dem UML-Klassendiagramm (siehe Abb. 9) mit Hilfe des Programms 'Enterprise Architect[6]' Quellcode erzeugt. Dieser Quellcode stellt die Grundlage für die weitere Programmierung dar.

Die generierten Klassen wurden durch weitere Funktionen, die nicht im Klassendiagramm dargestellt sind, erweitert. Das Meta Modell gibt ausschließlich, die Services bzw. Funktionen an, die der IEC 61850 Server zur Verfügung stellen soll. Die Datenstrukturen wurden hauptsächlich durch 'Getter'- und 'Setter'-Funktionen ausgestattet, um durch die baumartig aufgebauten Strukturen navigieren zu können und es so zu ermöglichen, die tatsächlichen Datenpunkte abzufragen. Eine Klasse hält eine Liste an Zeigern, die die unterliegenden Datenstrukturen referenzieren. Über die Funktionen, wie sie in Listing 1 dargestellt sind, kann beispielsweise auf das Datenattribut mit der Referenz "Bat/ZBTC.SoC.mag" zugegriffen werden. Da der Aufbau des Meta Modells so gestaltet ist, dass sich erst bei Instanziierung über SCL-Dateien entscheidet welche konkreten Datentypen tatsächlich von einem Datenattribut gehalten werden, ist je nach Datenstruktur eine Typumwandlung (Cast) zum eigentlichen Datentyp durchzuführen. Ist dies erfolgreich geschehen, kann über die entsprechenden Funktionen, die der Datentyp bereitstellt, auf den jeweiligen Datenpunkt zugegriffen werden. Wobei 'Server' ein Zeiger auf eine Instanz der 'GenServerClass'-Klasse ist. Um die konkreten Datenstrukturen über eine SCL-Datei zu erstellen, muss jene Datei zunächst ein 'Parsing' durchlaufen, in dem die für eine Instanziierung wichtigen, Informationen ermittelt werden. Für die Implementierung des Parsers wurde die *Xerces-C++ XML Parser API*[7] verwendet.

[6] `http://www.sparxsystems.de/uml/neweditions/`
[7] `http://xerces.apache.org/xerces-c/`

```
GenDataAttributeType *data;
data = Server->getLogicalNode("Bat", "ZBTC")->getDataObject
   ("SoC")->getData()->getAttribute("mag")->
   getAttributeType();

if (data->getType().compare("AnalogueValue") == 0) {
  AnalogueValue* aval = dynamic_cast<AnalogueValue*>(data);
  float value = aval->getValue();
  aval->setValue(20.092014);
}
```

Listing 1: Datenzugriff

Zum Speichern der Informationen ist für jedes SCL-Tag eine entsprechende Klasse angelegt worden. Diese Klassen speichern die jeweiligen Attribute des SCL-Tags. Nachdem über den Parser die verschiedenen Klassen erstellt wurden, kann anhand der gespeicherten Daten mit der Instanziierung der Datenobjekte begonnen werden. Für jede Klasse des Meta Modells ist definiert, aus welchen 'Unterklassen' sie aufgebaut ist. Diese Definitionen sind unter dem Tag '<DataTypeTemplates>' beschrieben. Welche Logischen Knoten und Devices tatsächlich instanziiert werden sollen ist unter dem Tag '<Server>' beschrieben. Für jedes Logische Device ist eine Menge an Logischen Knoten beschrieben. Innerhalb des Tags '<LN0>' bzw. '<LN>' kann ebenfalls festgehalten werden, welche Datenattribute bereits bei der Instanziierung mit entsprechenden Werten zu füllen sind. Für das Batteriesystem kann es sinnvoll sein den Batterietyp, den Hersteller oder den Betreiber anzugeben. Darüber hinaus können hier Data Sets definiert werden. Für den Anwendungsfall 'Einspeiseprognose einhalten' sind für den Logischen Knoten LLN0 zwei Data Sets definiert. Das Data Set 'Status' ist für das spätere Monitoring der Werte für die Ladespannung bzw. -strom sowie für den Ladestand (SoC) vorgesehen. Über das Data Set 'Steuern' sollen je nach Leistung der PV-Anlage die Werte für die Wirk- und Blindleistung (OutWSet, OutVarSet) vorgegeben werden. Anhand dieser Daten wurden zunächst die Instanzen für die verschiedenen Logischen Knoten erstellt. Diese Instanzen sind anschließend schrittweise erweitert worden, indem die entsprechenden Datenobjekte zunächst anhand der Typdefinitionen instanziiert und dann der Instanz des Logischen Knoten hinzugefügt wurden. Nach der Anfertigung der Datenstrukturen konnten über die Refrenzen der Datenattribute die bereits in der SCL-Datei definierten Datenwerte gesetzt werden. Um die für den Anwendungsfall nötigen Daten zu Speichern und Abrufen zu können, sind die Logischen Knoten LLN0, LPHD, ZBAT und ZBTC mit den jeweiligen Datenobjecten, Common Data Classes sowie Data Sets definiert und instanziiert worden.

Nachdem nun die Strukturen zum Halten der verschiedenen Datenwerte implementiert und erzeugt wurden, sind die Services 'GetDataSetValues' und 'SetDataSetValues' zum Datenaustausch sowie das Mapping auf das Manufacturing Messaging Specification Protokoll zu implementieren. Dazu wurden zwei weiteren Funktionen der Serverklasse hinzugefügt. Je eine Funktion zur Handhabung eines 'Read-Requests' und eines 'Write-Requests'. Sie stellen die Schnittstelle zwischen dem MMS Protokoll und den Services des IEC 61850 Standards dar. Ist ein Request empfangen worden, wird

zunächst festgestellt, um welche Art von Request es sich handelt. Es folgt zunächst die Prüfung der angegeben Data Set Referenz. Jene Referenz wird aufgelöst und somit geprüft, ob sich unter dieser Adresse tatsächlich ein DataSet befindet. Ist dem nicht so, wird ein entsprechender ServiceError an den Client zurückgeschicht. Ist die Referenz gültig, wird wie im Listing 1 gezeigt, auf das Data Set zugegriffen und es werden die verschiedenen DatenAttribute ausgelesen. Diese Werte werden anschließend in einem 'Read-Response' an den Client zurückgesendet. Handelte es sich um einen 'Write-Request', so wird ebenfalls auf das angegebene Data Set zugegriffen und die einzelnen mitgesendeten Datenpunkte über die entsprechenden 'Setter' auf dem Server gespeichert. Ist kein Fehler aufgetreten so wird dieser Vorgang mit dem Versenden eines 'Success' Flags quittiert.

```
while(1) {
  pvleistung = modbusVerbindung->readRegister();

  if (pvleistung < prognose)
    batterieVerbindung->SetDataSetValues
       ("Bat/LLN0.Steuern", wirkleistung, blindleistung);

  if (pvleistung > prognose)
    batterieVerbindung->SetDataSetValues
       ("Bat/LLN0.Steuern", wirkleistung ,blindleistung);

  batterieVerbindung->GetDataSetValues("Bat/LLN0.Steuern");
  Sleep(intervall);
}
```

Listing 2: Programmablauf Leitstelle

Die serverseitige Implementierung ist nun vollständig. Die benötigten Datenstrunkturen sind vorhanden und es kann auf eingehende Anfragen entsprechend reagiert werden. Damit jedoch eine tatsächliche Kommunikation umgesetzt werden kann fehlt noch die Implementierung der Leitstelle bzw. die clientseitige Implementierung. Wie eingangs bereits erwähnt und in der Abbildung 7 dargestellt, soll die Leitstelle über das Modbus Protokoll Leistungswerte von der (simulierten) PV-Anlage abrufen und je nach vorgebener Prognose die Leistung des Batteriesystems über den 'SetDataSetValues'-Service anpassen. Daneben sollen über den 'GetDataSetValues'-Service die Werte für den Ladestrom und die Ladespannung sowie der State-of-Charge überwacht werden. In Listing 2 ist der Ablauf im Leitsystem beispielhaft angegeben. Für das Versenden der Requests auf der Clientseite wurde der gleiche MMS Quellcode sowie das gleiche Mapping verwendet wie auf der Serverseite. Um die Modbus-Register des Modbus-Slaves abfragen zu können wurde die Bibliothek libmodbus[8] verwendet. Diese stellt sehr einfache Schnittstellen zur Verfügung, um mit wenig Aufwand eine Verbindung zwischen Leistelle und Modbus-Slave aufzubauen und damit einen Datenaustausch zu ermöglichen.

[8] http://libmodbus.org/

5.3 Zusammenfassung

In diesem Kapitel wurde die Realisierung des Anwendungsfalls ‘Einspeiseprognose einhalten’ erläutert. Zur Umsetzung des Anwendungsfalls bedurfte es zunächst der Abbildung des, im Konzept beschriebenen, Meta Modells in äquivalente C++-Klassen. Anschließend mussten den verschiedenen Klassen weitere Funktionen hinzugefügt werden, um eine Navigation und damit die Abfrage der konkreten Datenpunkte zu ermöglichen. Nachdem die Metaklassen vollständig programmiert waren, konnte über ein Parsing der SCL-Konfigurationsdatei (siehe Anhang C) die Instanziierung der einzelnen Datenstrukturen und -objekte realisiert werden. An die Programmierung der lokalen Datenstrukturen schloss sich die Umsetzung der Kommunikation zwischen Batteriesystem und Leitwarte sowie zwischen Leitwarte und simulierter PV-Anlage an. Umgesezt wurden die Data Set Services ‘GetDataSetValues’ und ‘SetDataSetValues’, in dem sie auf die entsprechenden Services des MMS Protokolls abgebildet wurden. Zum Datenaustausch zwischen Leitstelle und PV-Anlage bzw. ‘Modbus-Slave’ fand die libmodbus Bibliothek Anwendung.

Somit sind all jene Bausteine implementiert, um den Anwendungsfall ‘Einspeiseprognose einhalten’ zu realisieren. Anhand dieser Implementierung soll im folgenden Evaluierungskapitel das in dieser Arbeit entwickelte Konzept abschließend betrachtet sowie bewertet werden.

6 Evaluierung

In den vorangegangenen Kapiteln wurde ein Konzept zur Kommunikation mit Batteriespeichersystemen erarbeitet und anhand dieses Konzepts der Anwendungsfall 'Einspeiseprognose einhalten' prototypisch umgesetzt. Diese Implementierung dient als Grundlage für die im Folgenden beschriebene Evaluierung. Evaluiert wird das Konzept anhand der Umsetzung des Meta Modells, der Realisierung der Konfiguration sowie anhand des implementierten Anwendungsfalls.

6.1 Meta Modell

Grundlage des entwickelten Konzepts stellt das formal definierte Meta Modell (siehe Abbildung 9) dar. Es ist derart gestaltet, dass über Aggregationen die verschiedenen Klassen des Modells ineinander verschachtelt werden. Diese Aggregationen des UML-Klassendiagramms sind bei der Implementierung in C++ durch Zeiger auf die entsprechende Klasse realisiert. Durch 'Getter'- und 'Setter'-Funktionen kann auf die Objekte der Klassen zugegriffen werden. Alle in der Implementierung benötigten Datenpunkte konnten so abgerufen werden.

Aufgrund der starken Verschachtelung der Klassenstruktur, war der Zugriff auf die Datenattribute recht aufwändig. Zunächst musste anhand der Instanznamen die Referenz des Attributes aufgelöst werden. Da das Meta Modell sehr generisch gehalten ist, ist anschließend eine Typumwandlung (Cast) durchzuführen. War dies erfolgreich möglich, konnte der tatsächliche Datenpunkt abgerufen werden (siehe Listing 1). Dieser Aufbau der Klassenstruktur konnte in der Implementierung ohne Probleme umgesetzt werden und auch die Instanziierung der konkreten Objekte anhand des Meta Modells konnte wie konzipiert erfolgen. Damit ist die Anforderung an die Erhöhung der Interoperabilität zwischen Netzkomponenten erfüllt. Darüber hinaus ist durch das Modell die Trennung zwischen der Datenhaltung und der Kommunikationsstruktur möglich und so die Anforderung an die Kapselung der Softwarekomponenten ebenfalls erfüllt.

Um jedoch den Implementierungsaufwand sowie die Komplexität des Quellcodes zu verringern, ist die interne Datenhaltung nach Möglichkeit einfacher zu gestalten. Das bedeutet, dass die Verschachtelung der Klassenstruktur, so wie sie von der Norm vorgegeben wird, nicht zwingend auch in tatsächliche C++-Klassen umzusetzten ist. Es ist jedoch weiterhin die von der Norm IEC 61850 geforderte Unterteilung der Klassen den anderen Netzkomponenten mitzuteilen, um eine standardkonforme Kommunikation zu ermöglichen.

Neben dem generischen Meta Modell sind von der Norm ebenfalls Instanzen der abstrakten Klassen definiert, die die unterschiedlichen Funktionen der Netzkomponenten abbilden. Spezielle Instanzen die die Funktionen und Datenwerte eines Batteriesystems abbilden sind von der Erweiterung [IEC 61850-7-420] definiert. Der für ein Batteriesystem essentielle Datenpunkt, der den Ladestand abbildet, ist jedoch nicht in dieser Norm enthalten. Aufgrund der Möglichkeit weitere Instanzen dem Datenmodell standardkonform hinzuzufügen, konnte der Datenpunkt ohne weitere Probleme in

die Datenstruktur aufgenommen werden. Durch die Implementierung konnte weiterhin gezeigt werden, dass eine Erweiterung des Datenmodells unkompliziert möglich ist.

Zum Meta Modell gehören ebenfalls die abstrakten Services, die 'Requests' und 'Responses' für die Datenübertragung bereitstellen. Im Konzept sind diese Services als Funktionen der generischen Klassen definiert. Analog zur Überlegung zur internen Datenrepräsentation hat auch hier die Implementierung gezeigt, dass die Services nicht an die in der Norm beschriebenen Klassen gebunden sind. So kann beispielsweise die 'GenServerClass' die Verarbeitung der aller Services übernehmen. Wichtig ist diese Zuteilung von Services zu den Klassen, um feststellen zu können welche Anfrage an welche Datenstruktur gesendet werden kann. Wie intern am besten auf eine Service-Anfrage reagiert wird, könnte Bestandteil einer weiterführenden Arbeit sein.

Abschließend kann für das Meta Modell festgehalten werden, dass eine Eins-zu-eins-Umsetzung unter Umständen nicht die effektivste Methode darstellt, um intern Daten und Services zu organisieren, insbesondere im Hinblick auf eine Programmierung. Es war jedoch trotzdem möglich durch die Implementierung das entwickelte Konzept funktionstüchtig zu realisieren.

6.2 Konfiguration

Ein weiterer Teil des Konzepts ist die Konfiguration der konkreten Datenstruktur aus SCL-Dateien. Diese Konfiguration ermöglicht, je nach Anwendung, eine andere Datenrepräsentation zu erzeugen. Konzipiert wurde die Konfiguration als Teil der 'FileTransferClass'-Klasse. Nach der Übertragung einer SCL-Datei soll stets eine Neukonfiguration des Systems erfolgen. Als Randbedingung der Implementierung befindet sich hingegen die Konfigurationsdatei bereits auf dem Controller des Batteriesystems. Beim Start des Systems wird diese ausgelesen und die definierte Datenstruktur erzeugt.

Die Grundlage für die Instanziierung stellt das 'Parsing' der SCL-Datei dar. Aufgrund der verwendeten Bibliothek 'Xerces-C++' konnte das 'Parsing' der SCL-Datei effizient umgesetzt werden. Anhand der ausgelesenen Daten konnte Schritt für Schritt das Logical Device "Bat"bzw. der Server zusammengesetzt werden. Der Parser sowie die Logik zur Instanziierung der verschiedenen Datenobjekte ist nicht an den Einsatz in einem Batteriesystem gebunden sondern zur ebenso auch zur Erstellung anderer Datenstrukturen genutzt werden. Somit ist die Umsetzung der Konfiguration flexibel einsetzbar und es nicht wichtig welche Logical Nodes, DataObjects oder Common Data Classes für ein IED instanziiert werden sollen. Getestet wurde dies anhand von verschiedenen Ausprägungen der SCL-Datei aus Anhang C. Wobei die Anzahl und die Zusammensetzung der Logical Nodes variiert wurde. Als Randbedingung ist jedoch stets davon ausgegangen worden, dass die SCL-Datei keine Fehler, undefinierte Tags oder Attribute enthält.

Für die Konfiguration kann abschließend festgehalten werden, dass eine Implementierung durch die Programmiersprache C++ erfolgreich realisiert werden konnte und so die in der Anforderungsanalyse identifizierte Anforderung an eine flexible Datenstruktur erfüllt wurde.

6.3 Anwendungsfall 'Einspeiseprognose einhalten'

Im Folgenden soll nun die Evaluierung des Konzepts anhand der prototypischen Umsetzung des Anwendugsfalls 'Einspeiseprognose einhalten' erfolgen. Realisiert wurde die Kommunikation zwischen einer Leitstelle, einer PV-Anlage sowie einem Batteriesystem. Konkret handelt es sich um den Datenaustausch zwischen drei PC-Systemen, die die tatsächliche Kommunikation zwischen den zuvor genannten Netzkomponenten simuliert. Der Kommunikationsablauf der Testumgebung ist nachfolgend in Abbildung 8 veranschaulicht.

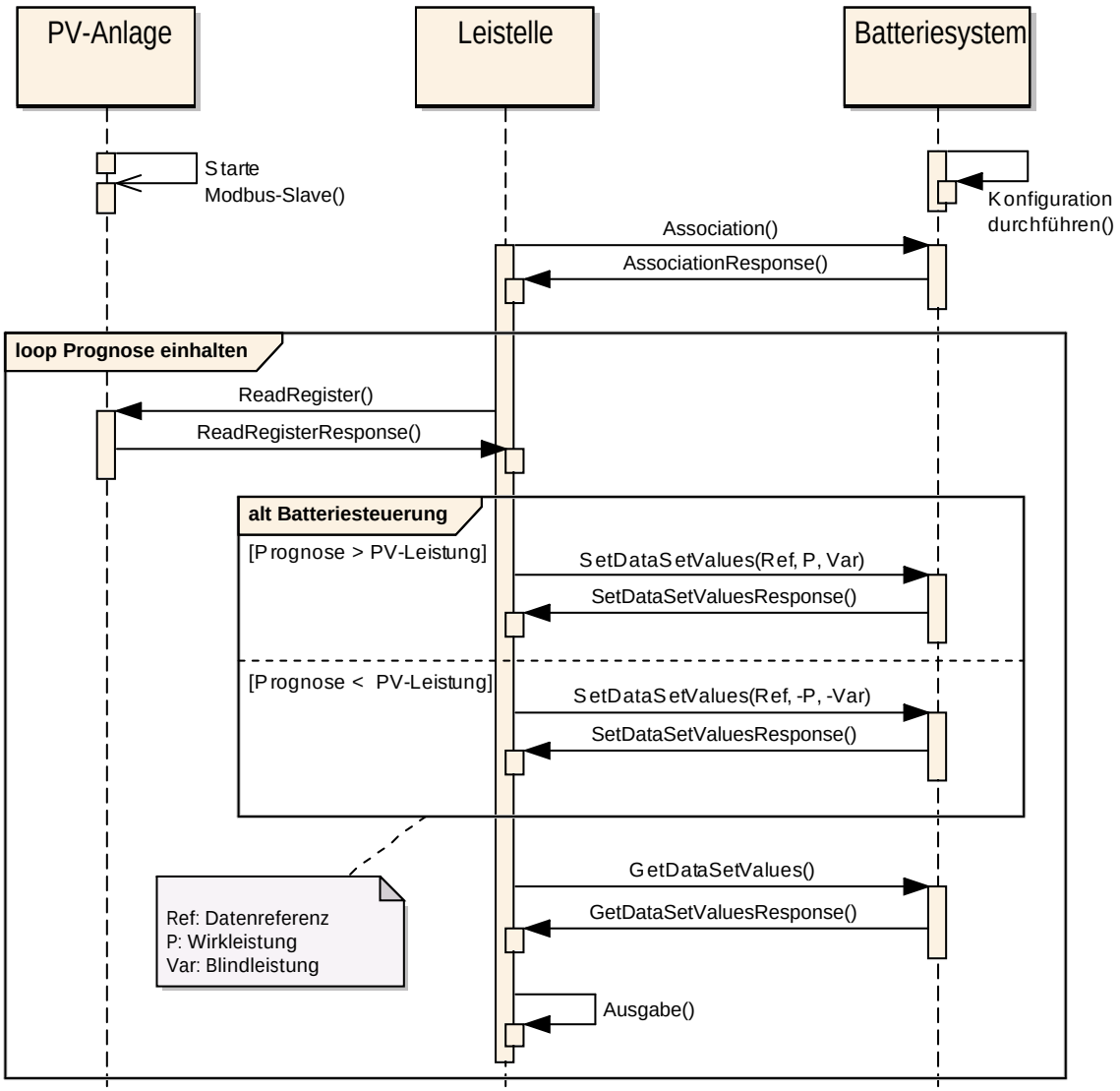

Abbildung 8: Kommunikationsablauf der Testumgebung

Zu Begin des Tests wird der 'Modbus-Slave' auf der Seite der PV-Ablage gestart und das Batteriesystem führt die Konfiguration des Datenschemas anhand der lokal vorliegenden SCL-Datei durch. Der 'Modbus-Slave' stellt die Leistungswerte der PV-Anlage zur Verfügung anhand deren das Leitsystem entscheidet, ob Strom zu laden oder einzuspeisen ist. Nachdem die Erzeugung der Datenstruktur durch das Batterie-

system erfolgreich beendet ist, wartet as System auf eine Verbindungsanfrage der Leistelle. Wurde diese korrekt empfangen wird an die Leitstelle eine positive Rückmeldung versandt. Anschließend wechselt die Leitstelle in eine Schleife, in der sie fortlaufend je nach Leistung der PV-Anlage die Wirk- und Blindleistung der Batterie anpasst und den Ladestrom, die Ladespannung und den aktuellen Ladestand der Batterie überwacht.

Zunächst wird von der PV-Anlage das entsprechende Register für die Leistung der Anlage abgefragt. Danach prüft die Leistelle ob die Anlage momentan mehr bzw. weniger Energie liefert als in der Prognose vorhergesagt wurde. Wird zu wenig Energie geliefert wird das Batteriesystem über den 'SetDataSetService' veranlasst die Werte für die Wirk- und Blindleistung entsprechend der Vorgabe der Leistelle anzupassen und so die fehlende Energiemenge ins Netz einzuspeisen. Wird mehr Energie geliefert als die Prognose vorgibt, wird durch das Setzen von negativen Leistungswerten die Batterie dazu veranlasst die überschüssige Energie zu speichern. Über den Service 'GetDataSetValues' werden anschließend die Attribute des Data Sets "Status" abgerufen. Über dieses Data Set kann der Ladestrom, die Ladespannung und der aktuelle Ladestand der Batterie abgefragt werden (siehe Anhang C). Abschließend werden die abgefragten Werte ausgegeben bzw. angezeigt.

Anhand dieses Testaufbaus konnte gezeigt werden wie eine mögliche Umsetzung des Konzepts anhand eines konkreten Anwendungsfalls aussehen kann. Es konnte das Zusammenwirken zwischen dem Meta Modell, der Instanziierung aus einer Konfigurationsdatei und der Datenübertragung über die Services evaluiert werden. Darüber hinaus konnten die nicht funktionellen Anfoderungen, die Übertragung von Messwerten sowie die Konfiguration erfüllt werden. Die Strukturen zum Senden eines Fahrplans sind dabei ebenfalls umgesetzt worden, jedoch ist in der Implementierung die Einhaltung der Einspeiseprognose über die Leistelle realisiert worden.

Anschließend lässt sich festhalten, dass die Norm IEC 61850 für den Einsatz mit Batteriespeichersystemen geeignet ist, wenngleich einige Anpassungen vorgenommen werden müssen. Diese Anpassungen schränken den Betrieb sowie die Kommunikation mit der Batterie jedoch nicht ein und sind standardkonform umsetzbar. Über die Norm ist somit eine vollständige Steuerung eines Batteriesystems realisierbar, die es ermöglicht das Batteriesystem im (intelligenten) Stromnetz effektiv zu betreiben.

7 Zusammenfassung und Ausblick

Abschließend soll noch einmal die gesamte Arbeit zusammengefasst werden und die wichtigsten Ergebnisse und Konzeptsdetails herausgestellt werden. Anschließend wird ein Ausblick auf mögliche weiterführende Arbeiten gegeben.

7.1 Zusammenfassung

In dieser Arbeit wurde ein Konzept zur Kommunikation für Batteriespeicher anhand der IEC Norm 61850 erarbeitet. Für die in der Anfoderungsanalyse identifizierten Anforderungen ließ sich diese Norm als am geeignetesten herausstellen. Konzipiert wurde ein Informationsmodell, welches aus einem generischen Datenmodell sowie verschiedenen Services zum Informationsaustausch besteht. Desweiteren ist durch Konfigurationsdateien die flexible Anpassung des Informationsmodell möglich.

Zunächst wurden im Kapitel der Anforderungsanalyse Geschäftsanwendungsfälle für den Einsatz von Batteriespeichern erarbeitet. Diese wurden in einem nächsten Schritt durch die Systemanwendungsfälle spezifiziert sowie Anforderungen an die Kommunikation herausgestellt. Anhand dieser Anforderungen konnten im Konzeptkapitel verschiedene Lösungsansätze diskutiert und bewertet werden. Die Norm IEC 61850 stellte sich dabei als am geeignetesten heraus, um diese Anforderungen an die Kommunikation zu erfüllen. Die Norm definiert ein umfassendes Informationsmodell, das neben einem Meta Modell und einer Menge an konkreten Datenklassen ebenfalls eine Konfigurationssprache beinhaltet, die es ermöglicht je nach Anwendungsfall eine Konfiguration des Datenschemas durchzuführen.

Ferner wurden im Konzept die notwendigen Datenklassen ausgewählt und entsprechend erweitert, um alle für ein Batteriesystem wichtigen Daten zu speichern und um ebenfalls ihre Semantik abbilden zu können. Zu den Datenklassen werden von der Norm ebenfalls abstrakte Services definiert, die eine Schnittstelle zwischen den Daten und der tatsächlichen Übetragngsstruktur bilden. Das 'Manufacturing Messaging Specification'-Protokoll ist als Übertragungsstruktur gewählt worden und ermöglicht den tatsächlichen Datenaustausch innerhalb eines Netzwerks. Darüber hinaus ist ein entsprechendes Mapping von den abstrakten Services auf das MMS definiert. Ferner wurde ein Konzept für Konfiguration der Datenstrukturen über entsprechende Konfigurationsdateien entwickelt.

Über die Implementierung des Anwendungsfalls 'Einspeiseprognose einhalten', der im Analysekapitel beschrieben wurde, konnte eine Evaluierung des Konzepts erfolgen. Es konnte gezeigt werden, dass das Konzept geeignet ist, um den Informationsaustausch zwischen einer Leitstelle und einem Batteriesystem zu realisieren.

7.2 Ausblick

Abschließend soll ein Überblick über mögliche Fragestellungen und Probleme gegeben werden, die Inhalt weiterer Forschungsarbeiten sein könnten.

Insbesondere im Hinblick auf die Informationssicherheit kann an vielen Stellen dieser Arbeit eine weiterführende Frage an die Sicherheit identifiziert werden. Das 'Control-Model', welches von der Norm IEC 61850 definiert wird beinhaltet bereits die Möglichkeit des "Select-Before-Operate" und die damit verbundenen Authentifizierungsmechanismen. Neben der Athentifizierung von Netzkomponenten ist auch die allgemeine Verschlüsselung der Informationsübertragung wichtig. Dies gilt insbesondere dann, wenn der Datenaustausch über ein öffentliches Netz wie beispielsweise das Internet abgewickelt werden soll. Ein Standard, der neben der Absicherung verschiedener Kommunikationsnormen ebenfalls Sicherheitsaspekte für die Norm IEC 61850 behandelt, ist der Standard IEC 62351.

Der Fokus dieser Arbeit liegt auf der Betrachtung der abstrakten Kommunikationsstrukturen und Datenmodelle. Damit ein tatsächlicher Einsatz der Kommunikation anhand der Norm 61850 ermöglicht werden kann, bedarf es der entsprechnden Serverlogik. Die Entwicklung einer solchen Logik könnte Gegenstand einer weiterführenden Arbeit sein.

Die in dieser Arbeit behandelte Fragestellung und Entwicklung des Konzepts ist eng an den Einsatz mit Batteriespeichern geknüpft. Die Kommunikation mittels der Norm IEC 61850 kann jedoch ebenso auch in Bereichen in denen andere Speichertechnologien eingesetzt werden Vorteile bringen, wie beispielsweise beim Einsatz in Pumpspeicherwerken.

Literaturverzeichnis

[Adr10] Adria, Oliver: Policy Recommendations for Energy Storage in Grids with a High Proportion of Renewable Energy. In: Styczynski, Zbigniew A. (Hrsg.) ; Lindemann, Andreas (Hrsg.): *Integration of Renewable Energies into the Grid*. Magdeburg : Zbigniew Antoni Styczynski and Andreas Lindemann, 2010 (Magdeburger Forum zur Elektrotechnik Band 35), S. 30–32

[AEW08] *Stromspeicher in der Energiewende – Untersuchung zum Bedarf an neuen Stromspeichern in Deutschland für den Erzeugungsausgleich, Systemdienstleistungen und im Verteilnetz*. Agora Energiewende (Smart Energy for Europe Platform (SEFEP) gGmbH), September 2014. – 050/10-S-2014/DE

[BS14] Buchholz, Bernd M. ; Styczynski, Zbigniew: *Smart Grids - Fundamentals and Technologies in Electricity Networks*. Springer-Verlag, 2014. – ISBN 978–3–642–45119–5

[DNP11] *Common Language for Distributed Storage Integration: Applying the DNP3 Application Profile for Smart Inverters*. EPRI, Palo Alto, CA, 2011. – 1023056

[DNP12] *PAP(Priority Action Plan)12: Mapping IEEE 1815 (DNP3) to IEC 61850 Objects*. NIST Smart Grid Collaboration Wiki for Smart Grid Interoperability Standards. `http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/PAP12DNP361850`. Version: 5.10.2012 18:06:40 von Ron Farquharson. – Zugriff: 10.02.2015 17:15

[Fab10] Fabian, Jürgen: Visionen zur elektrischen Energieübertragung zukünftiger europäischer Netze. In: Styczynski, Zbigniew A. (Hrsg.) ; Lindemann, Andreas (Hrsg.): *Integration of Renewable Energies into the Grid*. Magdeburg : Zbigniew Antoni Styczynski and Andreas Lindemann, 2010 (Magdeburger Forum zur Elektrotechnik Band 35), S. 7–9

[IEC 61850-6] Norm IEC 61850-7-6 Dezember 2009. *Communication networks and systems for power utility automation – Part 6: Configuration description language for communication in electrical substations related to IEDs*. – Edition 2.0

[IEC 61850-7-1] Norm IEC 61850-7-1 Juli 2003. *Communication networks and systems for power utility automation – Part 7-1: Basic communication structure for substation and feeder equipment - Principles and models*. – Edition 1

[IEC 61850-7-2] Norm IEC 61850-7-2 August 2010. *Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI).* – Edition 2.0

[IEC 61850-7-3] Norm IEC 61850-7-3 Dezember 2010. *Communication networks and systems for power utility automation – Part 7-3: Basic communication structure – Common data classes.* – Edition 2.0

[IEC 61850-7-4] Norm IEC 61850-7-4 März 2010. *Communication networks and systems for power utility automation – Part 7-4: Basic communication structure – Compatible logical node classes and data object classes.* – Edition 2.0

[IEC 61850-7-420] Norm IEC 61850-7-420 Juni 2009. *Communication networks and systems for power utility automation – Part 7-420: Basic communication structure – Distributed energy resources logical nodes*

[IEC 61850-8-1] Norm IEC 61850-8-1 Juni 2011. *Communication networks and systems for power utility automation – Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3.* – Edition 2.0

[JB03] Janssen, Dirk ; Büttner, Holger: EtherCAT – Der Ethernet-Feldbus - Funktionsweise und Eigenschaften – 1. Teil. In: *Elektronik: Fachmedium für industrielle Anwender und Entwickler* Vol. 52 (2003), Nr. 23, S. 64–73. – WEKA-Fachmedien

[Nau12] Naumann, André: *Leitwarte im Smart Grid*, Otto-von-Guericke-Universität Magdeburg, Diss., 2012. – ISBN 978-3-940961-81-5

[Oes09] Oestereich, Bernd: *Analyse und Design mit UML 2.3.* Oldenbourg Wissenschaftsverlag München, 2009. – ISBN 978–3–486–58855–2. – 9.,aktualisierte und erweiterte Auflage

[OSS13] Ortega, Alcides ; Shinoda, Ailton A. ; Schweitzer, Christiane M.: Performance Analysis of Smart Grid Communication Protocol DNP3 over TCP/IP in a Heterogeneous Traffic Environment. In: *2013 IEEE Colombian Conference on Communications and Computing (COLCOM)* (2013), Mai

[PPH+10] Pedersen, Anders B. ; Poulsen, Bjarne ; Hauksson, Einar B. ; holt, Chresten T. ; Andersen, Peter B. ; Gantenbein, Dieter: Facilitating a Generic Communication Interface to Distributed Energy Resources: Mapping IEC 61850 to RESTful Services. In: *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)* (2010), Oktober

[RMW04] Raynders, Deon ; Mackay, Steve ; Wright, Edwin: *Practical Industrial Data Communications – Best Practice Techniques*. Elsevier (Newnes), 2004. – ISBN 978–0–7506–6395–3. – 1. Auflage

[SIE14] *Siemens ist Trendsetter bei Prozessbus für Schutzgeräte im Übertragungsnetz*. Infrastructure & Cities Sector (Smart Grid Division) – Cigré, 24. bis 29. August 2014 in Paris, August 2014. – Pressemitteilung – `www.siemens.com/press/pi/ICSG201408051d`

[SW06] Schnell, Gerhard (Hrsg.) ; Wiedemann, Bernhard (Hrsg.): *Bussysteme in der Automatisierungs- und Prozesstechnik*. Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH, 2006. – ISBN 3–8348–0045–7. – 6., überarbeitete und aktualisierte Auflage

[Tie06] Tietze, Ernst-Günther ; Cichowski, Rolf R. (Hrsg.): *Netzleittechnik Teil 2: Systemtechnik*. VDE Verlag GmbH, VWEW Energieverlag GmbH, 2006. – ISBN 978–3–8022–0844–7. – 2. Auflage

[VDE08] *Smart Distribution 2020 – Virtuelle Kraftwerke in Verteilungsnetzen, Technische, regulatorische und kommerzielle Rahmenbedingungen*. VDE Verband der Elektrotechnik, Elektronik, Informationstechnik e.V., Juli 2008. – Studie der ETG im VDE

[VEU13] *Verordnung des Europäischen Parlaments und des Rates zu Leitlinien für die transeuropäische Energieinfrastruktur*. Amtsblatt der Europäischen Union, April 2013. – Verordnung (EU) Nr. 347/2013

A Anhang – Diagramme

A.1 Meta-Modell

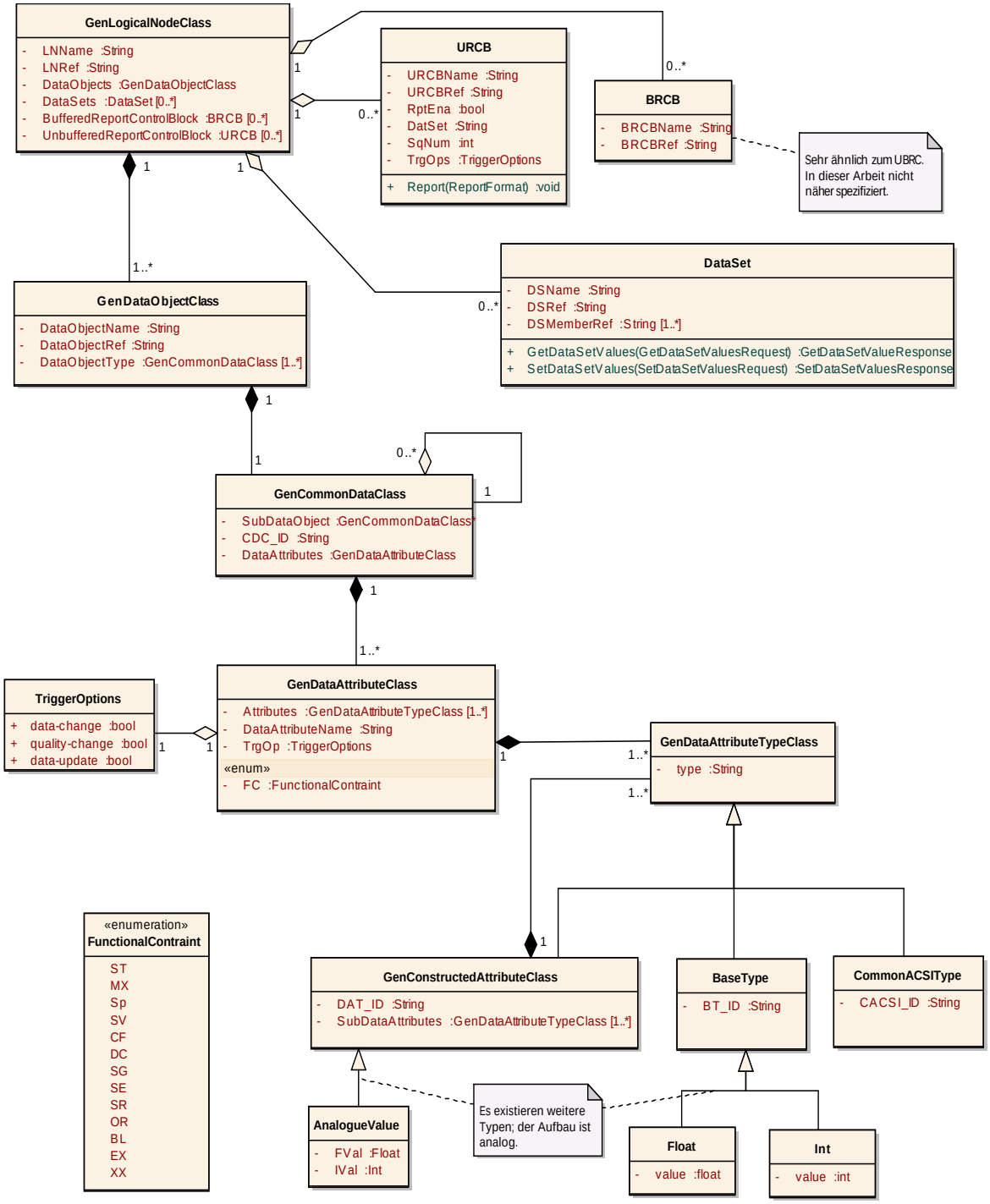

Abbildung 9: UML Klassendiagramm - IEC 61850 Meta Modell nach [IEC 61850-7-2]

A.2 UML Modell - Logging

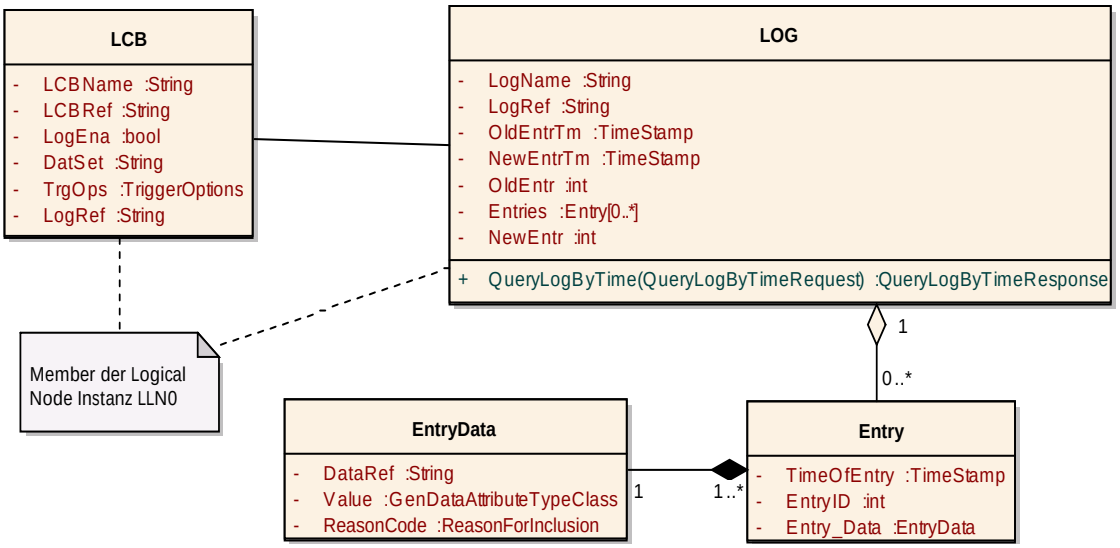

Abbildung 10: UML Klassendiagramm - Logging nach [IEC 61850-7-2]

A.3 UML Modell - Server und FileTransfer

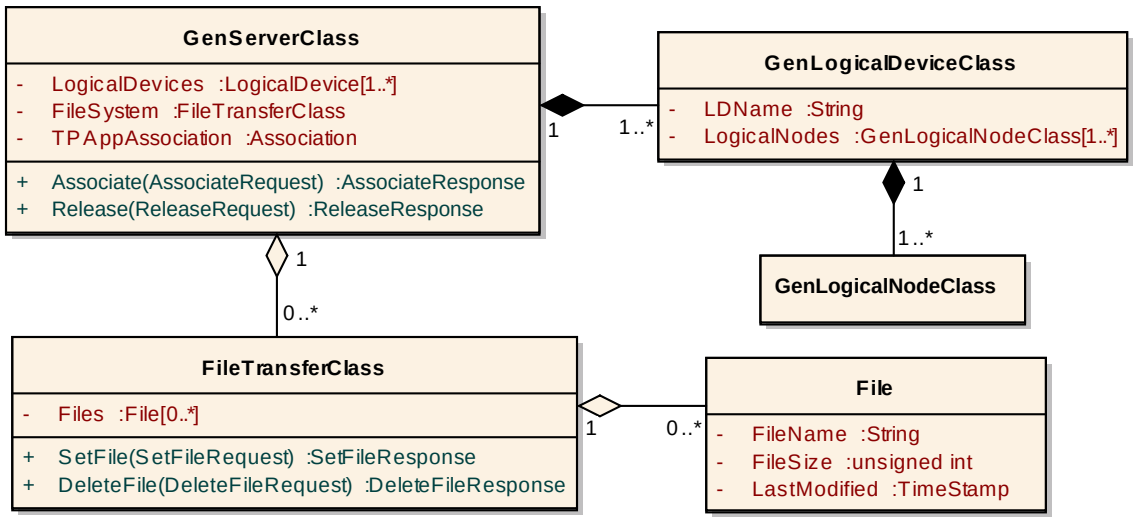

Abbildung 11: UML Klassendiagramm - Server und FileTransfer nach [IEC 61850-7-2]

B Anhang – Tabellen

B.1 Logical Nodes

Tabelle 16: Übersicht und Verbindung zwischen Informationen und Datenstrukturen nach [IEC 61850-7-4] und [IEC 61850-7-3]

Logical Node	CDC	Attribut	Information
LLN0	LPL	NamPlt	—
	ENS	Beh	—
	ENS	Health	Zustand
	ENC	Mod	Modus
LPHD	DPL	owner	Eigentümer
	ENS	PhyHealth	phys. Zustand
	SPS	Proxy	—
ZBAT	SPS	BatSt	Status (an/aus)
	ENG	BatTyp	Batterietyp
	CSG	DisChaCrv	Fahrplan
	ASG	AhrRtg	Kapazität
	MV	Vol	—
	MV	InBatV	Spannung
	MV	InBatA	Strom
	SPC	BatSt	Start/Stop System
	SPC	BatTest	Test starten
	SPS	BatTestRsl	Testergebnis
ZBTC	ENG	BatChaSt	—
	ENG	BatChaMod	—
	CSG	ChaCrv	Fahrplan
	MV	ChaV	Ladespannung
	MV	ChaA	Ladestrom
	MV	**SoC**	**Ladestand**
MMXU	MV	TotW	Wirkleistung
	MV	TotVAr	Blindleistung
	MV	Hz	Frequenz
MMDC	MV	Watt	Leistung
	MV	Amp	Strom
	MV	Vol	Spannung
MMTR	BCR	TotWh	Wirkenergie
	BCR	TotVArh	Blindenergie

Logical Node	CDC	Attribut	Information
ZINV	ASG	WRtg	—
	ENG	ACTyp	—
	CSG	PQVLimSet	—
	ASG	OutWSet	Wirkleistungssollwert
	ASG	OutVarSet	Blindleistungssollwert
	ASG	OutHzSet	Frequenzsollwert
DCCT	CUG	Currency	Währung
	ASG	CnttExpWLim	Energieexportlimit
	ASG	CnttImpWLim	Energieimportlimit
	CUG	OpCost	Betriebskosten pro Stunde
	CUG	OpWCost	Betriebskosten pro kWh
	CUG	StrCost	Startkosten
	CUG	StopCost	Stopkosten

B.2 Common Data Classes

Tabelle 17: Common Data Classes aus Tabelle 16

CDC	Name
ASG	Analogue setting
BCR	Binary counter reading
CSG	Curve shape setting
CUG	Currency setting group
ENC	Controllable enumerated status
ENG	Enumerated status setting
ENS	Enumerated status
DPL	Device name plate
LPL	Logical node name plate
SPC	Controllable single point
SPS	Single point status
MV	Measured value

B.3 MMS Mapping

Tabelle 18: Service Mapping

Service Name	Nachrichtentyp	MMS Service
Associate	Request	Initiate-Request Service
	–ServerAccessPoint	–Presentation Address
	Response+	Initiate-Response Service
	–AssociateID	–PresentationEndPoint
	Response−	Initiate-ErrorPDU Service
	–ServiceError	–MMS ServiceError
Release	Request	Conclude-Request Service
	–ServerAccessPoint	–Presentation Address
	Response+	Conclude-Response Service
	Response−	Conclude-ErrorPDU Service
	–ServiceError	–MMS ServiceError
Report	Report	MMS Information Report
	–RptID	–MMS String
	–Timestamp	–MMS Binary-Time
	–DataReference	–MMS String
	–Value	–MMS Data
	–Reason	–MMS Bit-String
DeleteFile	Request	FileDelete Request
	–FileName	–FileName
	Response+	FileDelete Response
	Response−	MMS ServiceError
	–ServiceError	–ServiceError
SetFile	Request	ObtainFile-Request
	–FileName	–Filename
	–FileData	FileRead-Request, FileClose-Request *
	Response+	FileRead-Response+ FileClose-Response+ *
	Response−	MMS ServiceError
	–ServiceError	–MMS ServiceError

* Der Inhalt einer Datei wird in mehreren Schritten mit ‘FileRead’ übertragen

Service Name	Nachrichtentyp	MMS Service
GetDataSetValues	Request	Read-Request Service
	–DataSetReference	–VariableAccessSpecification
	Response+	Read-Response Service
	–DataAttributValues	–ListOfAccessResult
	Response−	Read-Response Service
	–DataAttributValues	–ListOfAccessResult
SetDataSetValues	Request	Write-Request Service
	–DataSetReference	–VariableAccessSpecification
	–DataAttributeValues	–ListOfData
	Response+	Write-Response Service
	–Result	–Success
	Response−	MMS ServiceError
	–ServiceError	–ServiceError
QueryLogByTime	Request	MMS ReadJournal-Request
	–LogReference	–journalName
	–RangeStartTime	–startingTime (TimeOfDay)
	–RangeStopTime	–endingTime (TimeOfDay)
	Response+	MMS ReadJournal-Response
	–ListOfLogEntries	–ListOfJournalEnties
	Response−	MMS ServiceError
	–ServiceError	–ServiceError

Tabelle 19: Datenstruktur Mapping

ACSI Datenstruktur	MMS Datenstruktur
Server	Virtual Manufacturing Device
DataAttribute	Named Variable Object
Log	Journal Object
Boolean	Boolean
INT{8,16,32,64}{U}	Integer
FLOAT32	Floating-point
VISIBLE STRING	Visible-string
UNICODE STRING	MMS String
TimeStamp	MMS Binary-Time

INT{8,16,32,64}{U}: Integer mit verschiedene Bitlängen (signed/unsigned)
Visible-string: Ausschließlich ASCII Zeichen

C SCL - Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="SCL.xsl"?>
<SCL xmlns="http://www.iec.ch/61850/2003/SCL" xmlns:xsi="
   http://www.w3.org/2001/XMLSchema-instance" xsi:
   schemaLocation="http://www.iec.ch/61850/2003/SCL SCL.xsd
   ">
  <Header id="Battery Controller" nameStructure="IEDName"/>
  <Communication>
    <SubNetwork name="NONE">
      <ConnectedAP iedName="MyBattery" apName="P1">
        <Address>
          <P type="IP">192.168.171.42</P>
        </Address>
      </ConnectedAP>
    </SubNetwork>
  </Communication>
  <IED name="MyBattery" type="Battery" manufacturer="ADS-
     Tec" configVersion="1.0">
    <Services>
      <DynAssociate/>
      <GetDataSetValue/>
      <SetDataSetValue/>
      <ConfDataSet max="5"/>
      <FileHandling/>
    </Services>
    <AccessPoint name="P1">
      <Server>
        <Authentication none="true"/>
        <LDevice inst="Bat">
          <LN0  lnType="LLN0_0" lnClass="LLN0" inst="1">
            <DataSet name="Steuern">
              <FCDA ldInst="BT" lnClass="ZINV" lnInst="1"
                 doName="OutWSet" daName="setMag" fc="SP"/>
              <FCDA ldInst="BT" lnClass="ZINV" lnInst="1"
                 doName="OutVarSet" daName="setMag" fc="SP"
                 />
            </DataSet>
            <DataSet name="Status">
              <FCDA ldInst="BT" lnClass="ZBTC" lnInst="1"
                 doName="SoC" daName="mag" fc="MX"/>
              <FCDA ldInst="BT" lnClass="ZBTC" lnInst="1"
                 doName="ChaA" daName="mag" fc="MX"/>
              <FCDA ldInst="BT" lnClass="ZBTC" lnInst="1"
                 doName="ChaV" daName="mag" fc="MX"/>
```

```
      </DataSet>
      <DOI name="NamPlt">
        <DAI name="vendor">
          <Val>ADS-Tec</Val>
        </DAI>
        <DAI name="swRev">
          <Val>0.0</Val>
        </DAI>
        <DAI name="d">
          <Val>StoraXe System PJ-EST-000143</Val>
        </DAI>
        <DAI name="configRev">
          <Val/>
        </DAI>
      </DOI>
    </LN0>
    <LN  lnType="LPHD_0" lnClass="LPHD" inst="1">
      <DOI name="PhyNam">
        <DAI name="vendor">
          <Val>ADS-Tec</Val>
        </DAI>
        <DAI name="owner">
          <Val>Fraunhofer Gesellschaft IFF Magdeburg<
             /Val>
        </DAI>
      </DOI>
    </LN>
    <LN  lnType="ZBAT_0" lnClass="ZBAT" inst="1">
      <DOI name="BatTyp">
        <DAI name="setVal">
          <Val>Lithium</Val>
        </DAI>
      </DOI>
    </LN>
    <LN  lnType="ZBTC_0" lnClass="ZBTC" inst="1">
      <DOI name="SoC">
        <DAI name="mag">
          <Val>24.57</Val>
        </DAI>
      </DOI>
      <DOI name="ChaV">
        <DAI name="mag">
          <Val>400.0</Val>
        </DAI>
      </DOI>
      <DOI name="ChaA">
```

```
            <DAI name="mag">
              <Val>30.0</Val>
            </DAI>
          </DOI>
        </LN>
        <LN  lnType="ZINV_0" lnClass="ZINV" inst="1" />
      </LDevice>
    </Server>
  </AccessPoint>
</IED>
<DataTypeTemplates>
  <LNodeType id="LLN0_0" lnClass="LLN0">
    <DO name="Mod" type="ENC_0"/>
    <DO name="Health" type="ENS_0"/>
    <DO name="NamPlt" type="LPL_0"/>
  </LNodeType>
  <LNodeType id="LPHD_0" lnClass="LPHD">
    <DO name="PhyHealth" type="ENS_0"/>
    <DO name="PhyNam" type="DPL_0"/>
  </LNodeType>
  <LNodeType id="ZBAT_0" lnClass="ZBAT">
    <DO name="BatSt" type="SPS_0"/>
    <DO name="BatTyp" type="ENG_0"/>
    <DO name="AhrRtg" type="ASG_0"/>
  </LNodeType>
  <LNodeType id="ZBTC_0" lnClass="ZBTC">
    <DO name="ChaV" type="MV_0"/>
    <DO name="ChaA" type="MV_0"/>
    <DO name="SoC" type="MV_0"/>
  </LNodeType>
  <LNodeType id="ZINV_0" lnClass="ZINV">
    <DO name="OutWSet" type="ASG_0"/>
    <DO name="OutVarSet" type="ASG_0"/>
  </LNodeType>
  <DOType id="SPS_0" cdc="SPS">
    <DA name="stVal" fc="ST" bType="BOOLEAN" dchg="true"
      dupd="false" qchg="false"/>
    <DA name="q" fc="ST" dchg="false" dupd="false" qchg="
      true" bType="Quality"/>
    <DA name="t" fc="ST" dchg="false" dupd="false" qchg="
      false" bType="Timestamp"/>
  </DOType>
  <DOType id="ENC_0" cdc="ENC">
    <DA name="stVal" fc="ST" bType="Enum" dchg="true"
      dupd="false" qchg="false" type="Mod"/>
    <DA name="q" fc="ST" dchg="false" dupd="false" qchg="
```

```
      true" bType="Quality"/>
    <DA name="t" fc="ST" dchg="false" dupd="false" qchg="
      false" bType="Timestamp"/>
  </DOType>
  <DOType id="ENS_0" cdc="ENS">
    <DA name="stVal" fc="ST" bType="Enum" dchg="true"
      dupd="true" qchg="false" type="BatteryState"/>
    <DA name="q" fc="ST" dchg="false" dupd="false" qchg="
      true" bType="Quality"/>
    <DA name="t" fc="ST" dchg="false" dupd="false" qchg="
      false" bType="Timestamp"/>
  </DOType>
  <DOType id="LPL_0" cdc="LPL">
    <DA name="vendor" fc="DC" dchg="false" dupd="false"
      qchg="false" bType="VisString255"/>
    <DA name="swRev" fc="DC" dchg="false" dupd="false"
      qchg="false" bType="VisString255"/>
    <DA name="d" fc="DC" bType="VisString255"/>
    <DA name="configRev" fc="DC" dchg="false" dupd="false
      " qchg="false" bType="VisString255"/>
  </DOType>
  <DOType id="DPL_0" cdc="DPL">
    <DA name="vendor" fc="DC" dchg="false" dupd="false"
      qchg="false" bType="VisString255"/>
    <DA name="owner" fc="DC" dchg="false" dupd="false"
      qchg="false" bType="VisString255"/>
  </DOType>
  <DOType id="ENG_0" cdc="ENG">
    <DA name="setVal" fc="SP" bType="Enum" dchg="true"
      dupd="false" qchg="false" type="BatteryType"/>
  </DOType>
  <DOType id="ASG_0" cdc="ASG">
    <DA name="setMag" fc="SP" bType="AnalogueValue" dchg=
      "true" dupd="false" qchg="false"/>
    <DA name="units" fc="CF" bType="Struct" type="Unit"
      dchg="true" dupd="false" qchg="false"/>
  </DOType>
  <DOType id="MV_0" cdc="MV">
    <DA name="mag" fc="MX" bType="AnalogueValue" dchg="
      true" dupd="false" qchg="true"/>
    <DA name="q" fc="MX" dchg="false" dupd="false" qchg="
      true" bType="Quality"/>
    <DA name="t" fc="MX" dchg="false" dupd="false" qchg="
      false" bType="Timestamp"/>
  </DOType>
  <DAType id="Unit" iedType="">
```

```
    <BDA bType="Enum" name="SIUnit" type="SIUnit"/>
    <BDA bType="Enum" name="multiplier" type="multiplier"
      />
  </DAType>
  <EnumType id="Mod">
    <EnumVal ord="0">Off</EnumVal>
    <EnumVal ord="1">Island mode</EnumVal>
    <EnumVal ord="2">Online</EnumVal>
    <EnumVal ord="3">Error</EnumVal>
  </EnumType>
  <EnumType id="BatteryState">
    <EnumVal ord="0">off</EnumVal>
    <EnumVal ord="1">start-up</EnumVal>
    <EnumVal ord="2">balancing</EnumVal>
    <EnumVal ord="3">ready</EnumVal>
    <EnumVal ord="4">operating</EnumVal>
    <EnumVal ord="5">warning</EnumVal>
    <EnumVal ord="6">error</EnumVal>
  </EnumType>
  <EnumType id="BatteryType">
    <EnumVal ord="0">Not applicable / Unknown</EnumVal>
    <EnumVal ord="1">Lead-acid</EnumVal>
    <EnumVal ord="2">Nickel-metal hydrate(NiMH)</EnumVal>
    <EnumVal ord="3">Nickel-cadmium(NiCad)</EnumVal>
    <EnumVal ord="4">Lithium</EnumVal>
    <EnumVal ord="5">Carbon zinc</EnumVal>
    <EnumVal ord="6">Zinc chloride</EnumVal>
    <EnumVal ord="7">Alkaline</EnumVal>
    <EnumVal ord="8">Rechargeable alkaline</EnumVal>
    <EnumVal ord="9">Sodium sulphur(NaS)</EnumVal>
    <EnumVal ord="10">Flow</EnumVal>
    <EnumVal ord="99">Other</EnumVal>
  </EnumType>
  <EnumType id="SIUnit">
    <EnumVal ord="1"></EnumVal>
    <EnumVal ord="2">m</EnumVal>
    <EnumVal ord="3">kg</EnumVal>
    <EnumVal ord="4">s</EnumVal>
    <EnumVal ord="5">A</EnumVal>
    <EnumVal ord="6">K</EnumVal>
    <EnumVal ord="7">mol</EnumVal>
    <EnumVal ord="8">cd</EnumVal>
    <EnumVal ord="9">deg</EnumVal>
    <EnumVal ord="10">rad</EnumVal>
    <EnumVal ord="11">sr</EnumVal>
    <EnumVal ord="21">Gy</EnumVal>
```

```
      <EnumVal ord="22">q</EnumVal>
      <EnumVal ord="23">Celsius</EnumVal>
      <EnumVal ord="24">Sv</EnumVal>
      <EnumVal ord="25">F</EnumVal>
      <EnumVal ord="26">C</EnumVal>
      <EnumVal ord="27">S</EnumVal>
      <EnumVal ord="28">H</EnumVal>
      <EnumVal ord="29">V</EnumVal>
      <EnumVal ord="30">ohm</EnumVal>
      <EnumVal ord="31">J</EnumVal>
      <EnumVal ord="32">N</EnumVal>
      <EnumVal ord="33">Hz</EnumVal>
      <EnumVal ord="34">lx</EnumVal>
      <EnumVal ord="35">Lm</EnumVal>
      <EnumVal ord="36">Wb</EnumVal>
      <EnumVal ord="37">T</EnumVal>
      <EnumVal ord="38">W</EnumVal>
      <EnumVal ord="39">Pa</EnumVal>
      <EnumVal ord="40">K/s</EnumVal>
      <EnumVal ord="41">m^2</EnumVal>
      <EnumVal ord="42">m/s^2</EnumVal>
      <EnumVal ord="43">kg/m^3</EnumVal>
      <EnumVal ord="44">m^2/s</EnumVal>
      <EnumVal ord="45">W/m K</EnumVal>
      <EnumVal ord="46">J/K</EnumVal>
      <EnumVal ord="47">J/kg K</EnumVal>
      <EnumVal ord="48">VA</EnumVal>
      <EnumVal ord="49">Watts</EnumVal>
      <EnumVal ord="50">VAr</EnumVal>
      <EnumVal ord="51">cos(phi)</EnumVal>
    </EnumType>
    <EnumType id="multiplier">
      <EnumVal ord="-9">nano</EnumVal>
      <EnumVal ord="-6">micro</EnumVal>
      <EnumVal ord="-3">milli</EnumVal>
      <EnumVal ord="-2">centi</EnumVal>
      <EnumVal ord="-1">deci</EnumVal>
      <EnumVal ord="0"></EnumVal>
      <EnumVal ord="1">deca</EnumVal>
      <EnumVal ord="2">hecto</EnumVal>
      <EnumVal ord="3">kilo</EnumVal>
      <EnumVal ord="6">Mega</EnumVal>
      <EnumVal ord="9">Giga</EnumVal>
    </EnumType>
  </DataTypeTemplates>
</SCL>
```

Selbsständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit "Entwicklung und Umsetzung einer standardkonformen Kommunikation für Batteriespeicher" selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, sowie alle Zitate entsprechend kenntlich gemacht habe.

Magdeburg, den 26. März 2015

_____

Markus Wolf