FH Westküste
Wirtschaft und Technik

Datenverarbeitung und Visualisierung von Umrichter-Testbench-Daten

Jon Feddersen
22122017

Bachelorarbeit im Studiengang Elektrotechnik und Inforationstechnik

bei

Prof. Dr. rer. nat. Kristina Schädler

Semesteranschrift
Ochsendrift 31
25853 Drelsdorf

Studienfach
Elektrotechnik und
Inforationstechnik

Abgabetermin: 01.10.2025

Fachsemesterzahl: 9

Sperrvermerk

Diese Arbeit enthält vertrauliche Daten und Informationen des Unternehmens, in dem die Bachelor-/Masterarbeit angefertigt wurde. Sie darf Dritten deshalb nicht zugänglich gemacht werden.

Die für die Prüfung notwendigen Exemplare verbleiben beim Prüfungsamt und beim betreuenden Hochschullehrer.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DUTs Devices-Under-Test
XML Extensible Markup Language
DTD Document Type Definition
API Application Programming Interface
SAX Simple API for XML

1. Einleitung

Das Aufbereiten von Daten, sowohl in Bezug auf die Struktur, in der die Datensätze gespeichert werden, als auch in der visuellen Darstellung einzelner Datenreihen, ist für das Verstehen der dahinterliegenden Prozesse und der effizienten Arbeit mit ihnen unerlässlich. Das Erfassen von Messdaten und ihre Aufbereitung, so wie ihre Interpretation ist aus der modernen Welt nicht mehr wegzudenken. In allen Bereichen der Wissenschaft werden Messdaten erhoben, egal ob bei medizinischen Studien, Wetterdaten oder in der Industrie erhobenen Testdaten, Sie müssen alle verständlich aufbereitet werden.

Die Speicherung,Aufbereitung und Visualisierung von größeren Datenmengen wir heutzutage meisten mit Softwaretools durchgeführt. Aus diesem Grund gibt es eine Vielzahl von Anbietern die Lizenzen für solchen Softwaretools vertreiben. Diese Tools sind oft aber allgemein gehalten und biete daher für fachspezifische Bereiche nicht den passenden Aufbau und Funktionen, welches sich Effiziente, Umgänglichkeit und vor alledem in der Genauigkeit widerspiegelt.

Das Ziel dieser Arbeit ist es, eine Web-Applikation zu entwickeln, die als solch ein Softwaretool fungieren soll, welches spezifisch für einen Anwendungsbereich zugeschnitten ist. Die Applikation soll XML-Datensätze, die aus einem Umrichter-Teststand stammen strukturiert Speicher und visuell Darstellen kommen. Diese visuellen Daten sollen sowohl firmenintern zur Analyse als auch als in einem Kundenbericht nach außen weitergegeben werden, daher soll die Visualisierung sowohl einfach verständlich also auch professionell gehalten sein, hierbei soll möglichst auf eine intuitive und simple Nutzeroberfläche geachtet werden. Weiterhin soll die Applikation gut erweiterbar sein, um zukünftige Funktionen unkompliziert in die Software einzubetten. Zudem soll eine möglichst einfache Implementierung in die bestehende Softwareumgebung der Firma in dessen Rahmen diese Arbeite durchgeführt wird gewährleistet sein.

Um das gesetzt Ziel und die Anforderungen im Zeitrahmen dieser Arbeit bestmöglich zu erfüllen, wird nach dem Erlangen und Ausformulieren der theoretischen Grundlagen der Erstellung für den eigentlichen Entwicklungsprozess der Web-Applikation eine inkrementelle und iterative Vorgehensweise vorgesehen. Diese soll für Flexibilität und gutes Risikomanagement in Bezug auf die Abgabe sorgen, da selbst bei Komplikation eine funktionelle Version zur Abgabe bereitsteht. Dieser Ansatz zur Entwicklung der Software wird sich nur geringfügig auf den Aufbaue der wissenschaftlichen Arbeit auswirken, sie wird nur in den theoretischen Grundlagen und im Fazit behandelt werden.

Der allgemeine Aufbau des Hauptteils der Arbeit beginnt mit den Kapitel 2. Grundlagen, die sich mit den theoretischen Hintergründen zur Entwicklung der Applikation, sowie mit den einzelnen zu implementierenden Funktionen beschäftigt. Gefolgt von Kapitel 3. Analyse und Konzeption, welches sich mit dem bestehenden System und den einzufügenden Strukturen befasst. Das dritte Kapitel 4. Implemetierung beschreibt die Entwicklung der Web-Applikation. Kapitel 5. Integration und Test beschreibt die Imtegration in das bestehende System und das Testen der Funktionalität. Abschließend Kapitel 6. Fazit und Ausblick werden die Ergebnisse zusammengefasst und ein Ausblick auf mögliche Erweiterungen gegeben.

2. Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen für das Entwickeln der Software, so wie das notwendige Verständnis des Teststandes und seine Abläufe behandelt.

2.1. Überblick über den Umrichter-Prüfstand

In diesem Abschnitt wird der Umrichter-Teststand, von dem die zu verarbeitenden Datensätze stammen, beschrieben, da dies für das generelle Verständnis der einzulesenden Datensatzstruktur unerlässlich ist. Die genaue Bezeichnung des Teststandes „USTB DWT Test Bench (XCT0006-1)“ wird im Folgenden als Test-Bench oder Teststand benannt. Diese Art Test-Bench wird im Allgemeinen für die End-of-Line-Prüfung von unterschiedlichen Umrichtern nach ihrer Herstellung genutzt, um die Produktqualität und -funktionalität sicherzustellen. [1]

In dem hier vorliegenden Fall wird der Teststand verwendet, um die aus dem Feld kommenden Umrichter auf ihre weitere Nutzungstauglichkeit zu testen. Die weitere Nutzungstauglichkeit wird ermittelt, indem die Messwerte mit Mittelwerten, die von mehreren fabrikneuen Umrichtern stammen, verglichen werden. Diese Messwerte müssen sich in einem vorher definierten Toleranzbereich befinden, um weiter im Feld verwendet zu werden.

Die Umrichter werden in der gegebenen Fachliteratur zur Test-Bench als Devices-Under-Test (DUTs) bezeichnet. Diese Bezeichnung kommt auch in den Berichten auf dem Teststand vor, daher hat der Autor diese Abkürzung übernommen.

2.1.1. Aufbau des Teststandes

Die Test-Bench besteht aus mehreren Komponenten, die sich im Testraum in unterschiedlichen Schaltschränken befinden. Der Teststand besteht aus folgenden Hauptkomponenten: Die Test-Bench besteht aus mehreren Komponenten, die sich im Testraum in unterschiedlichen Schaltschränken befinden, und der Teststand umfasst folgende Hauptkomponenten:

- Das Netzteil wandelt die 400-V-Netzspannung in eine isolierte Gleichspannung für den Zwischenkreis um. Das Netzteil liefert maximal 80 kW mit 1200 V DC oder 800 V DC, welche Werte verwendet werden, kann vor Teststart bestimmt werden. In Abbildung 1 wird das Netzteil als PSU bezeichnet, was für „Power Supply Unit“ steht.
- Das Elektronik-Rack, auf dem die Mess- und Steuerkomponenten befestigt sind, ist ein wichtiger Bestandteil des Teststandes. Hier befindet sich auch der (XCS2100) System-Controller, der das ganze System mit dem PC, auf dem die Test-Bench-Software läuft, via Ethernet verbindet. In Abbildung 1 mit ER bezeichnet, für „Electronic Rack“.
- Der Testmatrix-Schrank, in dem die Sammelschienen für den Stromanschluss und die Schützen sitzen. In Abbildung 1 mit TM bezeichnet, für „Test Matrix cabine“.

- Der Schrank mit dem Kühlungssystem, da die Umrichter während des Betriebes mit Wasser oder Luft gekühlt werden müssen. In Abbildung 1 mit „Cool1“ bezeichnet.
- Der Carrier, auf dem die Umrichter befestigt werden, ist speziell für bestimmte Umrichter konstruiert. Dieser wird speziell für bestimmte Umrichter konstruiert. In Abbildung 1 wird der Carrier als Carrier 1 bezeichnet. Body text

Neben den Hauptkomponenten befinden sich außerhalb des Sicherheitsbereiches, der während des Betriebes nicht betreten werden darf, ein PC mit einer Software zum Steuern der Testeinrichtung sowie eine Betriebsanzeige und ein Notaus. [1]

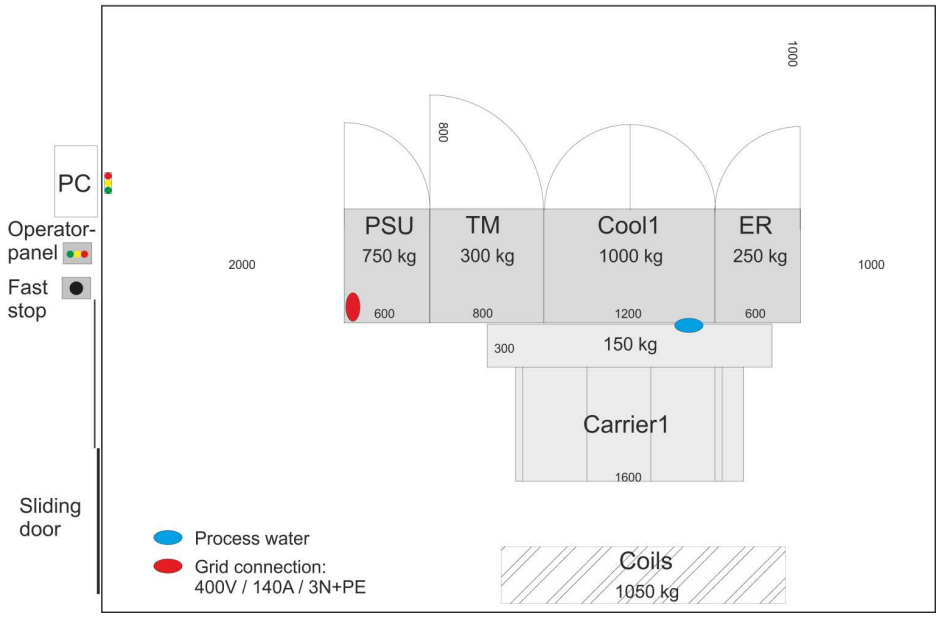

Abbildung 1: Aufbau des Teststandes
Quelle: [1, S. 7]

2.1.2. Testmodule

Es gibt mehrere Testmodule, die auf dem Teststand laufen und verschiedene Funktionen der Umrichter testen. Einige der Funktionen eines DUT können mit dem gleichen Modus eines Testmoduls überprüft werden, indem die entsprechenden Parameter ausgewählt werden. Jeder Testablauf ist autonom und kann mehrmals ausgeführt werden, auch mit unterschiedlichen Parametern.
Im Folgenden wird eine kurze Beschreibung der Funktionen der für diese Arbeit relevanten Testmodule gegeben.

Driver Consumption Test: Der Driver Consumption Test überprüft den Stromverbrauch des Treibers im Leerlauf und während Pulssprüngen.

Pulse Test: Der Impulstest verfügt über drei Funktionsmodi.

- Im Modus „Functional-Switching“ (FSW) kann überprüft werden, ob die Halbleiter generell schalten.

- Im Modus „Over-Current-Protection“ (OCP) kann die Überstromüberwachung weiche Kurzschlüsse überprüfen. Ein weicher Kurzschluss ist, wenn der Stromfluss nicht sofort und vollkommen unterbrochen wird.
- Im Modus „Dynamic-Short-Circuit-Protection“ (DSCP) wird ein harter Kurzschluss, also ein vollständiger und sofortiger Kurzschluss, überprüft.

Power Test: Mit diesem Test werden zwei verschiedene Funktionen getestet werden.
Der Burn-In-Test (BIT) dient dazu, die DUTs zyklisch zu betreiben und so reale Betriebszustände zu simulieren. Zudem kann mit Hilfe dieser Funktion die Kühltemperatur überprüft werden, um die korrekte Wärmeübertragung der Halbleiter sicherzustellen. Während des Übertemperaturschutz-Tests (OTP) wird ein DUT, ähnlich wie beim Burn-In-Test (BIT), nur mit reduzierter Kühlung betrieben, bis die maximal zulässige Kühlkörpertemperatur erreicht ist und die Temperaturschutzschaltung auslöst. [1]

2.1.3. Testablauf

Die Schrittfolge des Testablaufes mit Montage der DUTs ist klar festgelegt und vor jedem Durchlauf gleich. Die Montage läuft wie folgt ab:

1. Die Umrichter werden auf dem Carrier befestigt. Es werden meist 3 dieser Geräte gleichzeitig getestet, Abweichungen je nach Bauform. Die Reihenfolge der elektrischen Phasen ist bei Draufsicht des Carriers von links nach rechts U-V-W. Dies ist für das Layout des Berichtes relevant.
2. Der Kühlkreislauf wird angeschlossen, je nach Umrichtertyp Lüftungs- oder Wasserkühlung.
3. Die DUTs werden mit den AC- und DC-Link-Kontakten verbunden, um das Gerät zu betreiben und die DC-Spannung wieder abzuleiten.
4. Das Anbringen von Signalkabeln zwischen DUT und Merkurbox. Die Merkurbox dient als Schnittstelle zwischen DUTs und Test-Bench.
5. Das Montieren von VCE-Klemmen an die AC-Kontakte der DUTs.

Vor Beginn des Testablaufs benötigt die Test-Bench noch weitere Informationen zu den DUTs, Testsequenzen und der Hardware-Topologie. Diese Daten können über das Scannen von vorliegenden QR- oder Barcodes eingefügt werden. Alternativ können diese auch über ein Eingabefenster im Teststandprogramm eingetragen werden. Je nach Topologiekonfiguration kann auch ein zweiter Träger mit DUTs gescannt werden, Dies kommt bei dem Test, den das Unternehmen durchführt, nicht vor. [1]

Das eigentliche Testen mit den verschiedenen Testmodulen wird autonom durchgeführt. Im Unternehmen läuft der Test regulär wie folgt ab:

1. Durchführung des Driver Consumption Testes, um die Stromversorgung der Treiberplatine auf den Umrichter zu testen.

2. Durchführung des Pulse Testes im Modus Functional-Switching, dies prüft die Umrichter.
3. Durchführung des Power Testes, in den Berichten auch XPower Test genannt. Dieser Test simuliert die Belastung der Umrichter im Feld und überprüft, ob sich die Werte in einem bestimmten Toleranzbereich befinden.

Nach dem Durchlaufen eines Tests wird automatisch ein XML-Datenfile mit den erhobenen Messdaten generiert, die enthaltenen Messwerte und alle vorher bestimmten Einstellungen erstellt und in eine Datei eingefügt.

Bei der Demontage nach dem Testdurchlauf werden alle Schritte in umgekehrter Reihenfolge durchgeführt.

2.2. Verarbeitung von XML-Daten

Dieses Kapitel behandelt einige Grundlegende und für dies Arbeit relevante Aspekte von dem Dokumententype Extensible Markup Language (XML), da die Messwerte bzw. die Berichte die der Testand generiert im diesem Format vorliegen.

Bei XML handelt es sich um eine Auszeichnungssprachen, also eine formale Sprachen, die verwendet werden, um die Struktur und Darstellung von Daten oder Texten zu beschreiben [2]. XML wurde entwickelt, um Informationen in einem maschinenlesbaren und strukturierten Format zu speichern und zu übermitteln. Sie wird hauptsächlich in Bereichen wie Webdiensten, Datenbanken, Konfigurationsdateien eingesetzt. XML ermöglicht die hierarchische Organisation von Informationen in einem strukturierten Aufbau und kann sowohl für Menschen als auch für Maschinen interpretiert werden.

Das Grundkonzept hinter XML war, eine universelle einsetzbare und erweiterbare Sprache zu erschaffen, die von verschiedenen Systemen unabhängig von deren grundlegenden Technologieansatz genutzt werden kann. Hierbei ware das angestrebte Ziel Daten in einem einheitlichen Standard zwischen verschiedenen Anwendungen und Plattformen zu speichern und auszutauschen zu kömmen.

2.2.1. XML-Strukturaufbau

Eine XML-Datei beginnt mit Prolog, der die XML-Version und die verwendete Zeichencodierung definiert. In Abbildung 2 ist ein häufig genutzter Prolog dargestellt, der auch in den Teststand-Berichten genutzt wird. Die erste Zeile des Prologs ist die sogenannte XML-Deklaration. Die XML-Deklaration enthält häufig die Attribute sind version und encoding, jedoch nur das Attribut version ist Pflicht. Werden auch die anderen notiert, müssen sie in der angegebenen Reihenfolge deklariert werden. Attribut version Mit version wird die verwendete XML-Version angegeben. Das Attribut encoding gibt die im Dokument verwendete Zeichenkodierung an, d. h. mit welcher Codierung die Datei gespeichert wird. Fehlt die Angabe wird als Vorgabe UTF-8 (8-Bit Unicode Transformation Format) verwendet. Neben der XML-Deklaration können im Prolog auch noch

Verarbeitungsanweisungen und Verweise auf eine Document Type Definition (DTD) deklariert werden, dies sind jedoch optional und für diese Arbeit nicht weiter relevant. [3, S. 8, 9]

```
<?xml version="1.0" encoding="UTF-8"?>
```

Abbildung 2: XML Prolog Beispielcode
Quelle: eigene Darstellung

Der Hauptteil eines XML-Dokument besteht aus einer Reihe von Elementen, die durch Tags markiert sind. Für jedes Element gibt es ein Start- und ein EndTag, welcher das Element beginnt und beendent. Ein Starttag kann beispeißlwiese „<NamedesTags>“ so aus, dann würde der dazugehörige Endtag „</NamedesTags>“ so aussehen. Der entscheidende Unterschied ist hierbei der Schrägstrich beim Endtag. Der Name des Elementes wird durch den Inhalt der Keiler- und Großer-Zeichen bestimmt, bei diesem Beispiel wäre der Name „NamedesTags“. Elemente haben einen Inhalt der aus Text, weiteren Elementen oder aus beidem bestehen kann, wenn Elemente andere Elemente beinhalten werden Sie als Elterelemente und die enthaltenen Elemente oft Kindelemente bezeichnet. Diese Eigenschaft der Elementente sorg dafür das XML-Datein einer hierarchischen Baumstruktur folgen. Hierbei wird das oberste Element als Wurzelelement bezeichnet,im Englischen „root element“.[3, S. 10–14]

```
<buch>
  <titel>XML-Grundlagen</titel>
  <autor>Max Mustermann</autor>
</buch>
```

Abbildung 3: XML Elemente Beispielcode
Quelle: eigene Darstellung

Jedes Element kann neben Inhalt auch beliebig vielen Attributen ausgestattet sein, die zusätzliche Informationen enthalten. Attribute werden im Start-Tag eines Elements definiert, diese bestehen immer aus eine Attributnamen und einem Wert. Der Wert wird dabei mit Anführungszeichen deklariert, wie in Abbildung 4 gezeigt. [3, S. 10–14]

```
<buch genre="Lehrbuch">
  <titel>XML-Grundlagen</titel>
  <autor>Max Mustermann</autor>
</buch>
```

Abbildung 4: XML Attribute Beispielcode
Quelle: eigene Darstellung

Kommentare werden mit den Tags „<!– und „–>“ eingefügt und dienen der Dokumentation oder dem Hinweis auf bestimmte Teile des Codes [3, S. 10–14]. Dies ist für die automatisch generierten Berichte irrelevant, jedoch für das Beschreiben der Beispiele hilfreich. In Abbildung 3 und

Abbildung 4 werde diese zur Beschreiben verwendet. Kommentare werden beim Parsen des Dokuments ignoriert, parsen und Paser werden nachfolgenden Kapitel behandelt.

2.2.2. Verarbeiten von XML-Dateien

In diesem Abschnitt wird eine Zusammenfassung der grundlegenden Methoden und Techniken gegeben, um XML-Dateien mithilfe verschiedener Tools unabhängig von der verwendeten Programmiersprache zu verarbeiten. Es wird beschrieben, wie man XML-Dateien analysiert, modifiziert und überprüft, um sie für verschiedene Zwecke einsatzbereit zu machen.

Der erste Schritt beim verarbeiten einer acXML-Daten ist das Parsen. Hierbei wird die XML-Datei in ein Programm geladen und in ein Format umgewandelt, das das Programm interpretieren kann. XML-Paser prüfen hierbei auch die XML-Daten auf Korrektheit, also ob die Voralien eingehalten werden und das Dokument vollständig ist. Dabei wird in nicht-validierte Parser und validierende Paser differenziert. Der Unterschied besteht dadrin, dass validierte Paser neben der korrekte Schachtelung und Bezeichnung der Strukturelemente wie die nicht-validierten Paser auch noch auf eine Vorgabe einer Dokumenttypdefinition oder eines Schemas prüfen.[3, S. 10]

Paser werden verwendet um eine Applikation über eine Application Programming Interface (API) eine Schnittstelle auf ein XML-Dokument zu geben. Bei APIs wird in diesem Bereich zwischen zwei Grundtypen unterschieden:[3, S. 405]

Die baumbasierten APIs lesen über den XML-Parser das XML-Dokument ein, parst es und erzeugt ein Modell als Baum von Knoten im Arbeitsspeicher. Auf Grunde der im XML-Dokument vorkommenden Informationseinheiten wird in verschiedene Knotentypen unterschieden. Das generierte Modell dient der Applikation für die weitere anwendungsspezifische Verarbeitung. Ein Beispiel für eine baumbasierte APIs ist **DOM!** (**DOM!**).

DOM!ist Ein objektorientiertes Modell, das die Struktur eines XML-Dokuments abbildet. In der Baumstruktur wird das gesamte Dokument abgebildet, indem jedes Element, Attribut und jeder Text bzw. Inhalt als Knoten gilt. Das DOM hat den Vorteil, dass es das XML-Dokument vollständig im Arbeitsspeicher darstellt, was das Durchsuchen und Bearbeiten des Dokuments erleichtert. Zudem bittet es eine einfache Schnittstelle bereitstellt, um XML-Daten zu erreichen und zu verändern. Allerdings benötigt diese Herangehensweise viel Speicherplatz, da es das gesamte Dokument im Arbeitsspeicher ablegt, was bei großen XML-Dokumenten ein Problem darstellen kann.[3, S. 413, 414]

Die ereignisbasierten APIs lesen XML-Dokument sequenziell von begin durch ein und meldet während des Lesens jedes Ereignis durch sogenannte Callbacks an die aufrufende Applikation zurück. Ein Ereignis ist ein Signal, das Änderungen in dem Markup-Status anzeigt. Das Bedeutet Ereignise tritten bei Element-Tags, Zeichendaten, Kommentaren, Verarbeitungsanweisungen, sowie bei den Grenzen des Dokumentes auf. Der Parser sendet hierbei durch die Callbacks eine Mitteilung an die aufrufende Applikation, welche Ereignisse eingetreten sind. Das Programm, das den Parser aufgerufen hat, muss nun das Ereignis interpretieren und entsprechend reagieren. Das XML-Dokument wird hierbei nicht vollständig im Arbeitsspeicher gespeichert, sondern

in Teilen gelesen und bearbeitet. Deshalb eignen sich ereignisbasierte APIs besonders gut für die Verarbeitung großer XML-Dateien, da dies für eine geringe Arbeitsspeicherbelastung sorgt. Trotz der Effizienz hinsichtlich des Speicherverbrauchs sind ereignisbasierte APIs für die Datenmanipulation nicht sonderlich gut geeignet. Es wird nämlich durch das Teileweise einlesen keine umfassende Dokumentstruktur wie beim aufgebaut Die baumbasierten APIs, welches die Manipulation bzw. bearbeitung erschwert. Ein Beispiel für eine große ereignisbasierte API ist Simple API for XML (SAX). Diese API abreitet nach dem oben beschriebenen Prinzip und wurde ursprünglich als Java-API entwickelt, inzwischen gibt es SAX auch für weiter Sprachen wie z. B. C++, Perl und Python.[3, S. 405]

Neben diesen beide großen Beispielen gibt es auch einige kleiner Anbieter mit abgewandelten Methoden, die jedoch in dem Umfang deutlich kleiner gehalten sind und weniger Funktionen bitten. Ein Beispiel dafür wäre ElementTree. ElementTree ist eine unkomplizierte und minimalistische Python-Bibliothek, die für die Bearbeitung von XML genutzt wird. Sie präsentiert eine Baumstruktur, die das XML-Dokument effektiv darstellt und einfache Techniken zum Durchsuchen, Modifizieren und Erstellen von XML-Dokumenten bietet.[4]

2.3. Datenbankentwurf und Normalisierung

2.3.1. Einführung in Datenbanken

Datenbanken sind organisierte Sammlungen von Daten, die elektronisch gespeichert und verwaltet werden. Ihr primäres Ziel besteht darin, große Datenmengen strukturiert abzulegen, den Zugriff zu optimieren und Datenintegrität sicherzustellen. Im Gegensatz zu einfachen Dateisystemen bieten Datenbanken Mechanismen zur gleichzeitigen Nutzung durch mehrere Benutzer, zur Vermeidung redundanter Datenhaltung und zur effizienten Abfrage mittels spezieller Sprachen wie der Structured Query Language (SQL).

Die Entwicklung moderner Informationssysteme erfordert den Einsatz relationaler Datenbanken, dokumentenbasierter Systeme oder hybrider Modelle, um unterschiedliche Anforderungen an Konsistenz, Flexibilität und Skalierbarkeit zu erfüllen.

2.3.2. Prozess der Datenbankerstellung

Die Erstellung einer Datenbank umfasst mehrere aufeinanderfolgende Schritte, die sowohl technische als auch konzeptionelle Aspekte berücksichtigen:

1. Anforderungsanalyse
    - Ermittlung der fachlichen und technischen Anforderungen.
    - Identifikation der relevanten Datenquellen und Datenformate.
    - Festlegung von Integritäts- und Sicherheitsanforderungen.
2. Konzeptionelles Datenmodell
    - Erstellung eines Entity-Relationship-Modells (ERM) zur Abbildung der realen Welt in Entitäten, Attribute und Beziehungen.
    - Berücksichtigung von Kardinalitäten (1:1, 1:n, n:m).
3. Logisches Datenmodell
    - Überführung des konzeptionellen Modells in ein relationales Schema.
    - Definition von Tabellen, Spalten (Attribute), Primärschlüsseln und Fremdschlüsseln.
    - Festlegung von Datentypen und Constraints (z.B. NOT NULL, UNIQUE, CHECK).
4. Physisches Datenmodell
    - Umsetzung des logischen Modells in einer konkreten Datenbankmanagementsoftware (z.B. MySQL, PostgreSQL, Microsoft SQL Server).
    - Optimierung der Speicherstrukturen, Indexierung und Partitionierung.

5. Implementierung und Test
    - Erstellung der Tabellen und Relationen gemäß Datenbankschema.
    - Einfügen von Testdaten.
    - Überprüfung der Funktionalität und Performance.

2.3.3. Datenbankstruktur

Die Datenbankstruktur bezeichnet den Aufbau und die Anordnung der Datenelemente innerhalb eines Datenbanksystems. Bei relationalen Datenbanken basiert sie im Wesentlichen auf Tabellen und deren Beziehungen.

Wesentliche Elemente sind:

- Tabellen – Grundstruktur, in der Daten in Zeilen (Tupeln) und Spalten (Attributen) gespeichert werden.
- Primärschlüssel (Primary Keys) – eindeutige Kennzeichnung eines Datensatzes innerhalb einer Tabelle.
- Fremdschlüssel (Foreign Keys) – Verknüpfung zwischen Tabellen, um referenzielle Integrität sicherzustellen.
- Indizes – Datenstrukturen zur Beschleunigung von Abfragen.
- Views (Sichten) – virtuelle Tabellen, die auf Abfrageergebnissen basieren.
- Constraints – Regeln zur Sicherstellung der Datenintegrität (z.B. Wertebereiche, Pflichtfelder).

2.3.4. Normalisierung und Datenintegrität

Die Normalisierung ist ein methodischer Prozess zur Minimierung von Redundanzen und Anomalien. Sie erfolgt schrittweise in Normalformen (1NF, 2NF, 3NF, BCNF). Jede Normalform beseitigt spezifische Arten von Datenanomalien:

- 1. Normalform (1NF): Elimination mehrfacher Werte innerhalb einer Zelle; atomare Attribute.
- 2. Normalform (2NF): Entfernung partieller Abhängigkeiten.
- 3. Normalform (3NF): Entfernung transitativer Abhängigkeiten.

Datenintegrität umfasst die Sicherstellung von Korrektheit, Konsistenz und Vollständigkeit der Daten. Sie wird erreicht durch:

- Entity-Integrität (eindeutige Primärschlüssel).

- Referentielle Integrität (gültige Fremdschlüsselverweise).
- Domänenintegrität (gültige Wertebereiche und Datentypen).

2.4. Grundlagen der Datenvisualisierung

2.4.1. Begriff und Zielsetzung

Unter Datenvisualisierung wird die visuelle Aufbereitung von Daten verstanden, um Muster, Trends, Zusammenhänge oder Anomalien erkennbar zu machen. Sie stellt ein zentrales Hilfsmittel dar, um komplexe Informationen effizient zu kommunizieren und kognitive Verarbeitungsprozesse zu unterstützen[Shneiderman, 1996]. Durch die grafische Darstellung wird es dem Betrachter ermöglicht, große Datenmengen intuitiv zu erfassen, ohne ausschließlich auf numerische oder textuelle Formen angewiesen zu sein. Die Zielsetzung der Datenvisualisierung umfasst in der Regel drei Kernaspekte[Ware, 2021]:

1. Exploration – Unterstützung bei der Entdeckung neuer Zusammenhänge und Hypothesen.
2. Analyse – Erleichterung der detaillierten Untersuchung von Strukturen und Abhängigkeiten.
3. Kommunikation – Vermittlung von Ergebnissen an unterschiedliche Zielgruppen.

2.4.2. Theoretische Grundlagen

Die Grundlage wirksamer Datenvisualisierung liegt in der menschlichen Wahrnehmungspsychologie. Insbesondere die Gestaltungsgesetze (z.B. Gesetz der Nähe, Ähnlichkeit und Kontinuität) spielen eine zentrale Rolle, da sie bestimmen, wie Informationen visuell gruppiert und interpretiert werden [Wertheimer, 1923].

Zusätzlich beschreibt die Theorie der *kognitiven Belastung* (Cognitive Load Theory), dass Darstellungen so gestaltet werden sollten, dass sie die Arbeitsgedächtniskapazität nicht überlasten [Sweller, 1988].

Ein weiterer theoretischer Rahmen ist Shneidermans *Visual Information-Seeking Mantra*:

„Overview first, zoom and filter, then details-on-demand.“

Dieses Prinzip betont die Notwendigkeit einer schrittweisen Annäherung an Daten, um vom Gesamtüberblick bis hin zu spezifischen Details zu gelangen.

2.4.3. Formen der Datenvisualisierung

Datenvisualisierungen lassen sich in verschiedene Kategorien einteilen [Few, 2012]:

- Explorative Visualisierung: Interaktive Darstellungen, die zur Untersuchung und Hypothesengenerierung dienen.
- Explikative Visualisierung: Fokussierte, oft statische Darstellungen, die Ergebnisse gezielt kommunizieren.
- Interaktive Dashboards: Kombination mehrerer Visualisierungstypen, häufig für Monitoring und Entscheidungsunterstützung.

Je nach Datentyp und Analyseziel kommen unterschiedliche Diagrammformen und Techniken zum Einsatz:

- Zeitreihen: Liniendiagramme, Flächendiagramme
- Kategorische Daten: Balken- und Säulendiagramme
- Zusammenhänge: Streudiagramme, Blasendiagramme
- Verteilungen: Histogramme, Boxplots
- Hierarchien und Netzwerke: Baumdiagramme, Graphvisualisierungen

2.4.4. Qualitätskriterien

Eine gute Datenvisualisierung erfüllt folgende Kriterien[Tufte, 2001]:

- Klarheit: Vermeidung unnötiger grafischer Elemente („Chartjunk“)
- Genauigkeit: Wahrheitsgetreue Darstellung ohne Verzerrung von Skalen oder Proportionen
- Effizienz: Schnelle Erfassbarkeit der relevanten Information
- Ästhetik: Ansprechende Gestaltung zur Förderung der Akzeptanz
- Barrierefreiheit: Berücksichtigung farbsehschwacher Nutzer durch geeignete Farbpaletten

2.4.5. Technologische Aspekte

Mit dem Fortschritt moderner IT-Systeme haben sich leistungsfähige Werkzeuge und Bibliotheken für die Datenvisualisierung etabliert. Beispiele sind Matplotlib und Plotly im Python-Umfeld, D3.js im Webbereich sowie Business-Intelligence-Tools wie Tableau oder Microsoft Power BI.

In Webapplikationen ermöglichen interaktive Bibliotheken eine nahtlose Einbettung in Benutzeroberflächen, wodurch sowohl explorative als auch explikative Ziele unterstützt werden.

2.5. Anforderungen an modulare Softwareentwicklung

Die modulare Softwareentwicklung ist ein etabliertes Paradigma, das darauf abzielt, komplexe Softwaresysteme in klar abgegrenzte, wiederverwendbare und unabhängig voneinander entwickelbare Einheiten zu zerlegen. Ziel ist es, sowohl die Wartbarkeit, Erweiterbarkeit als auch die Qualität der Software zu erhöhen. Um diese Ziele zu erreichen, müssen spezifische Anforderungen an die Gestaltung und Umsetzung modularer Systeme beachtet werden.

2.5.1. Klare Modulabgrenzung und Verantwortlichkeiten

Ein Modul sollte eine klar definierte Aufgabe erfüllen und über eine eindeutig abgegrenzte Funktionalität verfügen. Diese Trennung wird in der Literatur häufig als *Separation of Concerns* (SoC) bezeichnet 111. Durch eine eindeutige Abgrenzung lassen sich Abhängigkeiten reduzieren, wodurch Änderungen in einem Modul nur minimale Auswirkungen auf andere Systemkomponenten haben.

2.5.2. Geringe Kopplung und hohe Kohäsion

Zwei zentrale Qualitätsmerkmale modularer Systeme sind niedrige Kopplung und hohe Kohäsion 222.

- Kohäsion beschreibt, wie eng die Elemente eines Moduls zusammenarbeiten, um eine spezifische Aufgabe zu erfüllen.
- Kopplung hingegen beschreibt den Grad der Abhängigkeit zwischen einzelnen Modulen. Ein hoher Kohäsionsgrad bei gleichzeitig niedriger Kopplung erleichtert sowohl die Wiederverwendung als auch die Wartung.

2.5.3. Einheitliche Schnittstellen

Module interagieren über klar definierte und stabile Schnittstellen. Diese sollten standardisiert, dokumentiert und möglichst unabhängig von internen Implementierungsdetails sein 333. Eine wohldefinierte API (Application Programming Interface) ermöglicht die parallele Entwicklung mehrerer Module und erleichtert zukünftige Erweiterungen.

2.5.4. Wiederverwendbarkeit

Ein wesentliches Ziel der Modularisierung ist die Wiederverwendbarkeit von Modulen in unterschiedlichen Projekten oder Kontexten. Wiederverwendbare Module reduzieren Entwicklungsaufwand, erhöhen die Konsistenz und tragen zur Qualitätssteigerung bei 444. Hierfür ist eine generische und konfigurierbare Implementierung erforderlich.

2.5.5. Erweiterbarkeit und Anpassungsfähigkeit

Modulare Software muss so konzipiert sein, dass neue Funktionen ohne weitreichende Änderungen am bestehenden System integriert werden können. Dieses Prinzip wird oft mit dem *Open/Closed Principle* aus den SOLID-Prinzipien beschrieben: „Software entities should be open for extension, but closed for modification“ 555.

2.5.6. Testbarkeit

Durch die Abgrenzung von Modulen wird eine gezielte und unabhängige Prüfung einzelner Systemkomponenten möglich. Unit-Tests, Integrationstests und Mocking-Strategien profitieren stark von einer modularen Struktur 666. Testbare Module führen zu einer höheren Softwarequalität und geringeren Fehlerquoten im Betrieb.

2.5.7. Konsistenz und Standardisierung

Die Implementierung sollte einheitliche Konventionen in Bezug auf Namensgebung, Dokumentationsstandards und Code-Struktur aufweisen. Konsistenz erleichtert die Zusammenarbeit in Entwicklerteams und minimiert Missverständnisse 777.

2.5.8. Technologische Unabhängigkeit

Ein modularer Aufbau sollte es ermöglichen, einzelne Komponenten ohne tiefgreifende Änderungen an andere Plattformen, Technologien oder Bibliotheken anzupassen. Dies reduziert technologische Abhängigkeiten und verlängert die Lebensdauer der Software 888.

3. Analyse und Konzeption

3.1. Defintion der Anforderungen

3.2. Analyse der bestehenden Strukturen und Prozesse

3.3. Datenbankdesign und Strukturkonzeption

3.4. Grundkonzept des Benutzeroberflächen-Design

3.5. Entwurf der Applikationsarchitektur

4. Implementierung

4.1. Einlesen und Verarbeiten von XML-Daten

4.2. Implementierung der Datenbank

4.3. Entwicklung der Benutzeroberfläche

4.4. Technische Details zur Visualisierung

5. Integration und Test

5.1. Einbindung in die bestehende Systemlandschaft

5.2. Testmethoden und Durchführung

5.3. Ergebnisse der Testmethoden

6. Fazit und Ausblick

6.1. Zusammenfassung der Ergebnisse

6.2. Kritische Bewertung

6.3. Möglichkeiten für zukünftige Erweiterungen

i. Literaturverzeichnis

Literatur

[1] A. GmbH, *USTB DWT (XCT0006-1) Main Manual V1.0*, Aachen: AixControl GmbH, 2018.

[2] P. D. G. Neumann, „Auszeichnungssprache,“ in *Gronau, Norbert ; Becker, Jörg ; Kliewer, Natalia ; Leimeister, Jan Marco ; Overhage, Sven (Herausgeber): Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon*, [Online; Stand 18. August 2025], Berlin : GITO, 2019. Adresse: https://wi-lex.de/index.php/lexikon/technologische-und-methodische-grundlagen/sprache/auszeichnungssprache/.

[3] M. Becher, „XML-Grundlagen,“ in *XML: DTD, XML-Schema, XPath, XQuery, XSL-FO, SAX, DOM*. Wiesbaden: Springer Fachmedien Wiesbaden, 2022, ISBN: 978-3-658-35435-0. DOI: 10.1007/978-3-658-35435-0_1. Adresse: https://doi.org/10.1007/978-3-658-35435-0_1.

[4] Python. „xml.etree.ElementTree — The ElementTree XML API. “Adresse: https://docs.python.org/3/library/xml.etree.elementtree.html#module-xml.etree.ElementTree.

ii. Anhangsverzeichnis

iii. Erklärung

Hiermit erkläre ich, dass ich die von mir eingereichte Bachelor- / Masterarbeit "......(Titel der Arbeit).....ßelbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.
Ort und Datum

persönliche Unterschrift

(Name des Verfassers)