Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Darstellung der Topologie eines industriellen Kommunikationsnetzwerks

Master-Abschlussarbeit

Betreuer: Prof. Dr. Georg Rock und Frank Fährmann

Hattersheim, 22.6.2018

Vorwort und Danksagung

Die vorliegende Masterarbeit ist im Rahmen meiner Tätigkeit als Junior Softwareentwickler in der Firma Hilscher Gesellschaft für Systemautomation mbH (im Folgenden Firma Hilscher genannt) entstanden. Ich war in der Abteilung Software User Interface beschäftigt. Die meiner Thesis zugrundeliegende Software wurde in Zusammenarbeit mit dem Entwicklerteam des Engineering Tools Communication Studio entwickelt, da sie im letzteren als Plugin[1] verwendet werden soll.

An dieser Stelle möchte ich mich bei der Firma Hilscher für die Möglichkeit bedanken, in einem interessanten Themenfeld zu arbeiten. Ganz besonderer Dank gilt Herrn Frank Fährmann für eine umfassende, fachliche Betreuung meiner Arbeit.

Auch möchte ich bei allen Kollegen, die ich im Rahmen meines Projektstudiums interviewt habe, danken. Ohne ihre Kooperation hätte ich meine Arbeit nicht realisieren können.

Darüber hinaus gebührt jedem mein Dank, der durch Korrekturen, Verbesserungsvorschläge und hilfreiche Diskussionen zur Qualität dieser Arbeit beigetragen hat.

Mein größter Dank gebührt meiner neugeborenen Tochter und meiner Frau, die mir in dieser Zeit eine unersetzbare seelische und moralische Stütze waren.

[1] Plugin ist ein Softwaremodul in der Softwaretechnik, das eine bestehende Software erweitert, um spezifische Aufgaben zu erledigen.

Kurzfassung

Im Kontext der industriellen Kommunikation, bei der Konfiguration von Systemen mit unterschiedlichen Gerätetypen und Protokollen werden für das durchgängige Engineering geeignete Modelle benötigt. Der Schwerpunkt liegt hierbei auf die Visualisierung bzw. Überwachung der Systeme und der unterlagerten Netzwerk-Topologien. Für diese Zwecke hat die Firma Hilscher auf Basis einer vorhandenen Desktop- bzw. Webanwendung ein Engineering Tool mit dem Namen Communication Studio entwickelt. Die Nutzung dieser Software ist jedoch in puncto Topology Engineering umständlich in der Bedienung.

Im Rahmen dieser Arbeit soll ein Topology Engineering Plugin für das Communication Studio als eine Web-Komponente in HTML5 entwickelt werden. Die grafische Oberfläche soll es hierbei ermöglichen, einzelne Funktionen der Visualisierung abstrakt und intuitiv zu nutzen. Des Weiteren sollen einzelnen Darstellungsszenarien abgespeichert und geladen werden können, die protokollneutral die Topologien industrieller Kommunikationsnetzwerke ermitteln und darstellen.

Eine automatisierte Anordnung der Geräte und deren Verbindungen soll ebenfalls zur Verfügung stehen. Besonderes Augenmerk sollte hierbei auch auf eine einfache zukünftige Erweiterbarkeit der Software mit neuen Funktionen gelegt werden.

Im ersten Schritt der Arbeit soll anhand einer Analyse der Protokolle PROFIBUS und PROFINET IRT mögliche Topologiemuster ermittelt werden. Auf dieser Basis soll ein Konzept zur Topologie-Erkennung und -Darstellung entwickelt werden. Abschließend soll das Konzept in die Praxis umgesetzt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Problemstellung

Unter Kommunikation versteht man den Austausch von Daten zweier Einheiten. Diese Einheiten können zwei Softwaremodule sein, die sich auf einem Gerät oder auch auf unterschiedlichen Geräten befinden. In einem Netzwerk können sich mehrere Module oder Geräte befinden, die Daten miteinander austauschen. Neben der Darstellung des logischen Datenflusses gibt es auch eine physikalische Darstellung der Netzwerke ohne Berücksichtigung des Datenflusses. In dieser Arbeit soll nur die physikalische Darstellung, d. h. die Verknüpfung der Geräte eines industriellen Netzwerks betrachtet werden. In der industriellen Kommunikation kommen nun noch weitere Anforderungen hinzu wie z.B. Echtzeitverhalten und Übertragungssicherheit. Grundsätzlich unterstützt die Firma Hilscher alle gängigen Netzwerkprotokolle, die in der Fabrikautomation eingesetzt werden. In den vergangenen Jahren ist eine Vielzahl neuer, Ethernetbasierten Protokolle hinzugekommen. Durch die permanent steigenden Anforderungen an den industriellen Netzwerken steigt auch der Aufwand für ihr Management im Allgemeinen, insbesondere für ihre Konfiguration und Inbetriebnahme. Durch die geringe Anzahl an Netzwerkprotokollen, konnte man früher davon ausgehen, dass sich der Inbetriebnehmer eines Netzwerkes mit der Konfiguration auskannte. Heute muss man davon ausgehen, dass der Anwender kein detailliertes Wissen über die Konfiguration des Netzwerkes besitzt. Aus diesem Grund ist es umso wichtiger die Konfiguration für den Anwender einfach und intuitiv zu gestalten. Ein weiterer wichtiger Grund ist die Reduzierung des Support-Aufkommens, der sich aus einer leicht zu bedienenden, intuitiver Software ergibt.

Zur Konfiguration eines Netzwerkes gehören noch weitere protokollabhängige Parameter und Einstellungen, wie z.B. die Stationsadresse des Gerätes und die Übertragungsgeschwindigkeit der Daten, die an dieser Stelle nicht betrachtet werden und als gegeben vorausgesetzt werden.

Die Visualisierung der Netzwerktopologie ermöglicht es, dem Anwender einen schnellen und leichten Überblick sowie Einstieg in das Netzwerk zu geben. Mit dieser Arbeit sollen die Grundlagen für einen vereinheitlichten Topologie-Editor geschaffen werden, damit nicht für jedes entstehende industrielle Netzwerkprotokoll ein neuer Topologie-Editor programmiert werden muss. Ziel dabei ist, in einem Topologie-Editor verschiedene Netzwerke darzustellen. Für die Firma Hilscher sind, neben dem Vorteil der Kostenreduzierung und der Wiederverwendbarkeit der Softwarekomponente, das einheitliche „*Look and Feel*“ der Software für den Kunden von großer Bedeutung.

1.1 Motivation

Die Firma Hilscher sucht eine Möglichkeit, das Engineering Tool Communication Studio um eine grafische Komponente für die Abbildung komplexer, heterogener industriellen Netzwerke zu erweitern. Das grafisch unterstützte Topology Engineering bietet folgende Vorteile:

- Der Mensch kann diese Art der visuellen Informationen am schnellsten erfassen und verstehen
- Die Wiederverwendbarkeit dieser Komponente spart den Aufwand und Resources ein
- reduziert Engineeringzeit für Netzwerkinbetriebnahme und ermöglicht somit Kostenreduktion für den Endkunden.

Der aktuelle Stand des Communication Studio bietet grafische Darstellung Netzwerktopologien nur für zwei Netzwerktypen, die

- unhandlich sind,
- protokollabhängig in jeweiliger Binärkomponente verankert sind,
- auf verschiedene Technologien basieren und schwer zu warten und zu erweitern sind.

Gefordert ist ein protokollneutraler, einheitlicher Ansatz für alle von Hilscher unterstützten Protokolle. Die entstehende Komponente soll als ein Plugin in die Software Communication Studio eingebettet werden.

1.2 Zielsetzung

Das Ziel der vorliegenden Arbeit ist die Konzeption, das Design und die Prototypenentwicklung einer Topology Engineering-komponente als Plugin für das Communication Studio, die

- es dem Anwender einfaches Management der verschiedenartigen Netzwerke erlaubt,
- protokollneutral implementiert ist,
- alle gängigen Topologiemuster im Kontext der industriellen Netzwerke unterstützt,
- leicht erweiterbar ist, um künftige Netzwerke zu unterstützen.

Dabei stellt sich die Forschungsfrage, mit welchen Technologien lässt sich eine Client-Anwendung und eine Server-Anwendung mit einer möglichst einheitlichen Codebasis umsetzen, mit dem Ziel, Entwicklungs- und Wartungsaufwand möglichst gering zu halten und bei der bestehenden Software Communication Studio, bei der eigentlichen Host-Applikation, vernachlässigbare strukturelle Veränderungen vorauszusetzen.

1.3 Aufgaben

Durch in Abschnitt 1.2 vorgegebene Ziele ergeben sich nun folgende Aufgaben im Rahmen dieser Masterarbeit. Es sollen zwei verschiedene Software-Komponenten entstehen, die Server- und die Clientkomponente. Der Server kapselt die protokollabhängigen Netzwerkinformationen und stellt diese einheitlich dem Client zur Verfügung. Die Clientkomponente soll dann die Topologien der verschiedenartigen industriellen Kommunikationsnetzwerke unabhängig vom Netzwerkprotokoll darstellen und die Bearbeitung in einem zulässigen Rahmen im jeweiligen Kontext zulassen.

Da dieses Themengebiet sehr komplex und umfangreich ist und viel Fachwissen voraussetzt, erfolgt im ersten Schritt eine Einführung und Einarbeitung in das Thema. Danach sollen anhand der Protokolle PROFIBUS und PROFINET IRT, Vorgabe durch die Aufgabenstellung, verschiedene Topologiemuster und ihre Darstellungsarten analysiert werden. Auf Basis der durch die Evaluierung gewonnenen Erkenntnisse wird im zweiten Schritt ein Konzept

zur Darstellung der unterschiedlichen Topologien entwickelt. Nach der Erstellung des Konzeptes erfolgt dann die Implementierung der jeweiligen Komponenten in mehreren Iterationsstufen.

Für eine bessere Handhabe und Kontrolle der Masterarbeit, wird das Projekt in verschiedene Phasen mit entsprechenden Meilensteinen eingeteilt.

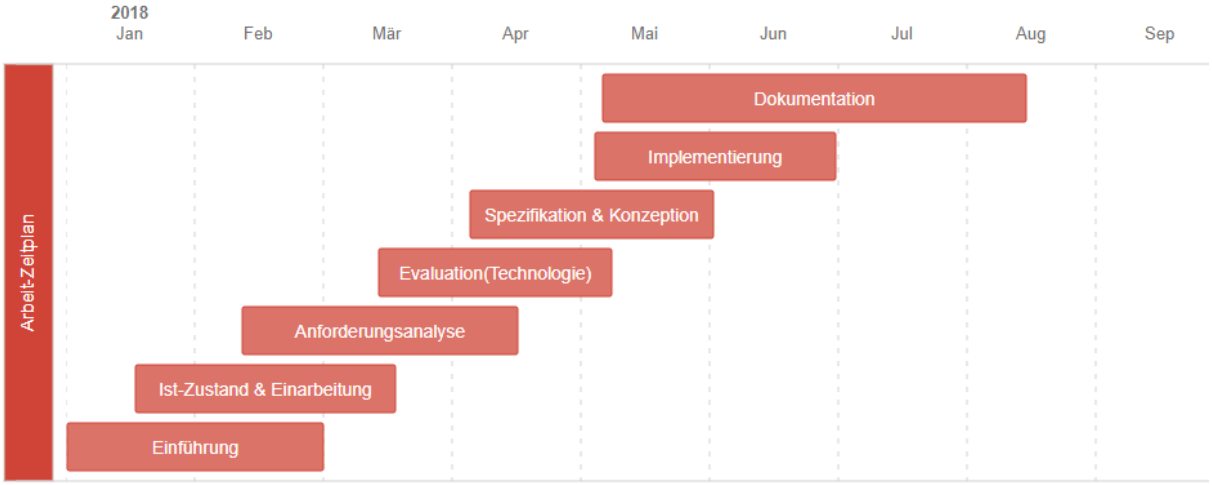


Abbildung 1: Zeitplan für die Entwicklung von Plugin Topology-Editor

In der nachfolgenden Tabelle sind diese Projektphasen in Detail aufgeschlüsselt.

Phase	Beschreibung	Ressource	Meilenstein
Einführung	Erklärungen und Erläuterungen der Aufgaben	Projektleiter gemeinsam mit allen Beteiligten	Ziele der Arbeit analysieren
Ist-Zustand und Einarbeitung	Analyse von Communication Studio und Einarbeitung der theoretischen Grundlage	Student	Topologien kennenlernen, Protokolle und Host-Applikation Communication Studio einarbeiten
Anforderungsanalyse	Anforderungen der Anwender untersuchen	Student	Funktionale und nichtfunktionale Anforderung auflisten
Evaluation	Analyse verschiedenen Frameworks	Student	Frameworks auswählen
Spezifikation und Konzeption	Erstellung einer Spezifikation und Konzeption	Student mit Gruppenleiter	Ansichten spezifizieren und Softwaremodule konzipieren
Realisierung	Erstellung des Prototyp	Student Entwicklungsumgebung VS 2017	Prototype erstellen
Dokumentation	Erstellen von Text und Grafiken	Student Word, Drucker	Schriftlicher Teil der Masterarbeit erstellen

Tabelle 1: Meilensteinplanung für die Masterarbeit

Die Terminplanung für Meilensteine und Fertigstellung des Projektes erfolgte in Abstimmung mit dem Entwicklerteam der Software Communication Studio.

1.4 Firmenvorstellung

Die Firma Hilscher Gesellschaft für Systemautomation mbH ist ein Familienunternehmen mit Hauptsitz in Hattersheim am Main. Die Gründung des Unternehmens erfolgte 1986 durch Hans-Jürgen Hilscher. Die Geschäftsführer sind Hans-Jürgen Hilscher und Sebastian Hilscher. Die Firma beschäftigt heute über 260 Mitarbeiterinnen und Mitarbeiter an 10 Standorten weltweit.

Durch klare Organisationsstrukturen, kundengerechte und qualitätsorientierte Leistungserfüllung ist das Unternehmen ein verlässlicher Partner für die Kunden auf der ganzen Welt.

Nach aktuellsten Feldbus- und Real-Time-Ethernet-Standards bietet die Firma eine Vielzahl Kommunikationslösungen an, nämlich:

- PC[2]-Karten
- Gateways
- OEM[3]-Aufsteckmodule für die Automatisierungs- und Visualisierungslösungen
- ASICs basierte netX-Technologie (Hilscher Gesellschaft für Systemautomation mbH, 2016)
- Protokollstacks
- Softwarelösungen für Konfiguration, Inbetriebnahme und Diagnose, Analyse der Netzwerke
- Industrie 4.0 / IIoT[4]-Lösungen (z.B.: Edge-Gateways mit Cloudanbindungslösungen)

Diese werden weltweit von namhaften Automatisierungsherstellern erfolgreich eingesetzt.-Im Bereich der Entwicklungsdienstleistungen und kundenspezifischer Baugruppen-Fertigung ist die Firma ein anerkannter Systempartner der großen Hersteller (z.B. B&R Industrial Automation GmbH, Schneider Electric SE, Stäubli) und zählt eine Vielzahl von Ingenieurbüros und Systemintegratoren zu seinen Kunden.

Abbildung 2: Haupteingang Hilscher Gesellschaft für Systemautomation mbH

[2] PC steht für Personal Computer
[3] OEM steht für Original Equipment Manufacturer
[4] IIoT steht für Industrial Internet of Thing

2 Grundlagen

In diesem Kapitel wird eine kurze Einführung in die für die vorliegende Arbeit wichtige Grundlagen gegeben. Eingegangen wird auf die der Arbeit zugrundeliegenden Basis- bzw. Standardtechnologien, die Fachbegriffe und Zusammenhänge. Des Weiteren folgt eine Analyse der Netzwerke und deren Strukturen. Abschließend folgt eine kurze Einführung in die Netzwerk-Protokolle PROFIBUS und PROFINET IRT und eine Übersicht über ihre Funktionsweise.

2.1 Grundlegende Begrifflichkeiten

Die Grundlage für das Erreichen der oben genannten Ziele, siehe Abschnitt 1.2, ist eine klare Definition und Zusammenfassung von wichtigen Begriffen, insbesondere für die industrielle Kommunikation. Diese sind wie folgt:

- **Netzwerk** (Plenk, 2017, S. 3) - als Netzwerk bezeichnet man den Verbund mehrerer Geräte oder Gerätegruppen zum Zweck der Datenkommunikation. Die wesentlichen Bestandteile der meisten Netzwerke sind Server und Clients. Das größte Netzwerk ist heutzutage das Internet.
- **TCP** (Heinrich Hübscher, 2011, S. 315) - über TCP werden verbindungsorientierte Datenübertragungen zwichen den Kommunikationspartnern (Sender und Empfänger) ermöglicht. Das bedeutet, dass während der Datenübertragung permanent spezifische Meldungen ausgetauscht werden.
- **IP (Internet Protocol)** (Heinrich Hübscher, 2011, S. 315) - es ist in der Informationstechnologie weit verbreitetes Netzwerkprotokoll. Im Vergleich mit TCP werden hier verbindungslose Datenübertragungen eingesetzt. Das bedeutet, dass keine dauerhalte Verbindungen zwischen den beiden Kommunikationspartnern besteht. Jeder Rechner im Internet besitzt eine eindeutige IP-Adresse, die einem Host-Namen entspricht.
- **Ethernet-Technik** (IT Wissen Info, 2001) - unter Ethernet versteht man eine Datenübertragungstechnik, die es ermöglicht, Daten innerhalb geschlossener Netzwerke zwischen verschiedenen Geräten zu übertragen. Da sie für lokale Datennetze konzipiert ist, spricht man von LAN-Technik.
- **Netzwerkprotokoll** (Olbrich, 2003, S. 84) - unter Netzwerkprotokoll versteht man ein Kommunikationsprotokoll für den Informationsaustausch zwischen den Teilnehmern in einem Netzwerk. Damit Kommunikationspartner überhaupt miteinander kommunizieren können, müssen bestimmte Vereinbarungen und Regeln eingehalten werden. Diese Regeln sind die Protokolle.

- **OSI-Referenzmodell** (Heinrich Hübscher, 2011, S. 274) - das OSI[5]-Referenzmodell besteht aus sieben Schichten und wird daher auch als 7-Schichten-Modell bezeichnet. Jede Schicht besitzt Funktionen und Protokolle, die spezifischen Aufgaben bei der Kommunikation zwischen zwei Systemen erfüllen müssen.
- **Feldbus** (KUNBUS GmbH, 2016) - unter Feldbus versteht man ein leitungsgebundenes Bussystem, das es ermöglicht, Daten in einer Prozessautomatisierungstechnik verschiedener Sensoren und Aktoren zu einem übergeordneten Steuerungsgerät zu übertragen. Eine Vielzahl von Bussystemen ist heute im Einsatz. Die Systeme unterscheiden sich in der Netzarchitektur, Datenübertragungsrate, Datenübertragungsmenge und der Übertragungstechnik.
- **Topologie** (Alexandroff, Paul, Hopf, Heinz, 1945, S. 2) - in Allgemein beschreibt man Topologie als Teilgebiet der Mathematik, welches ursprünglich diejenigen Eigenschaften geometrischer Gebilde behandelt, die bei umkehrbar eindeutigen, stetigen und totalen Abbildungen erhalten bleiben. In der industrielleren Kommunikation ist die Topologie laut (Schnabel, 2018) die Verbindungen der teilnehmenden Geräte untereinander, die den Datenverkehr ermöglichen.
- **Controller** (Popp, 2016, S. 381) - ein Controller ist die übergeordnete Steuerung, die typischerweise das Prozessabbild und das Anwenderprogramm enthält. Er ist der aktive Teil bei der die Kommunikation der angeschlossenen Devices parametriert und konfiguriert wird. Er führt den zyklischen/azyklischen Datenaustausch sowie die Alarmbearbeitung mit den prozessnahen Devices durch.
- **Supervisor** (Popp, 2016, S. 33) - dies kann ein Programmiergerät, Personal Computer oder Human-Machine-Interface-Gerät zu Inbetriebsetzungs- oder Diagnosezwecken sein. Um beispielsweise den Prozess für Testwecke zu steuern oder die Diagnoseauswertung zu übernehmen, kann ein Supervisor temporär die Funktionsweise eines Controllers übernehmen.
- **Device** - ein Device ist eine passive Station bei der Kommunikation, die gemäß Protokoll die Prozessdaten an die übergeordnete Steuerung überträgt und kritische Anlagenzustände beispielerweise Diagnose und Alarme meldet.
- **Knoten & Kanten -** Knoten repräsentieren die Kommunikationspartner und die Kanten die Verbindungen in der industriellen Kommunikation.
- **Logische Topologie** (IT Wissen Info, 2001)- eine logische Topologie beschreibt den Weg, den Daten nehmen, um die Kommunikation in einem Netzwerk zu gewährleiten.
- **Physikalische Topologie** (IT Wissen Info, 2001) **-** eine physikalische Topologie bezieht sich auf den Aufbau eines Netzes, d.h. in welcher Struktur die einzelnen Kommunikationspartner miteinander verbunden sind. Man kann diese Topologie durchaus mit einer Landkarte vergleichen. Die logische und physikalische Topologien müssen nicht identisch sein.

2.2 Grundlegende Netzwerktopologien

In diesem Abschnitt werden kurz die bekanntesten Topologien vorgestellt. Für die verschiedenen Netzwerketopologien werden die Vor- und Nachteile aufgezählt. An den unterschiedlichen

[5] OSI steht für Open Systems Interconnection

Vor- und Nachteilen der Topologie kann man auch schon erkennen für welches Einsatzgebiet diese Systeme verwendet werden können.
Topologien werden grafisch mit Knoten und Kanten dargestellt. In großen Netzen findet man oftmals eine Struktur, die sich aus mehreren verschiedenen Topologien zusammensetzt.
An dieser Stelle sei noch einmal erwähnt, das logische Topologie nicht Bestandteil dieser Arbeit sind, sondern nur die physikalische Topologie betrachtet wird.

2.2.1 Linien-Topologie

Die Linien-Topologie war einer der ersten Netze, die in kleineren Firmen und privaten Hauhalten zu finden war. Ein Koaxialkabel bildet die Basis, die Linie, welches beliebig, bis zu einen gewissen Grad, verlängert werden kann. Die Stationen werden über T-Stücke[6] an das Kabel angeschlossen. Die angeschlossenen Stationen greifen die Signale vom Kabel ab und senden wo sich das Signal dann in Richtung ausbreitet Am Anfang und Ende des Kabels sind in der Praxis Widerstände angebracht um Reflexionen[7] zu verhindern. Die Linien-Topologie wird auch als Bus-Topologie bezeichnet. Abbildung 3 veranschaulicht diese Topologiemuster mit sieben Stationen.

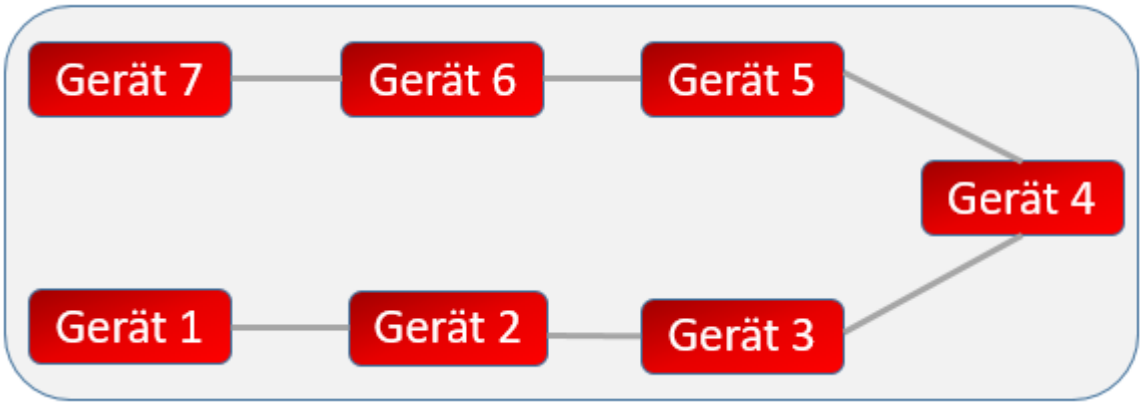


Abbildung 3: Linien-Topologie

Vorteile

- Das System gewährleistet möglichst geringe Kosten, da nur geringe Kabelmengen erforderlich sind
- Die Anschlüsse über T-Stücke sparen zum erheblich Aufwand, da nur eine einfache Verkabelung nötig ist
- Das Netzwerk ist schnell und einfach zu erweitern und dadurch speziellen Kundenwünschen präzise anzupassen
- Es werden keine weiteren Geräte zur Übermittlung der Daten benötigt
- Der Ausfall einer einzelnen Station hat für die Funktionalität des übrigen Netzes keine Konsequenzen

Nachteile

- Die Übertragung von Daten über ein einziges Kabel ist deutlich langsam
- Ein defektes Kabel an einer einzigen Stelle im Netz behindert den gesamten Netzwerkstrang

[6] T-Stücke sind Verbindungsglieder, die drei Anschlüsse besitzen
[7] Reflektionen sind ungewollte elektrische Wellen, die auf eine Übertragungsleitung entstehen.

- Es kann immer nur eine Station Daten senden. Während der Sendung sind alle anderen Stationen blockiert, dadurch entsteht den Datenstau
- Aufgrund der Möglichkeit der Datenkollisionen[8] sollte das Medium nur zu ca. 30% ausgelastet werden
- Datenübertragungen können leicht abgehört werden dadurch entsteht ein Sicherheitsproblem

2.2.2 Ring-Topologie

Eine Ring-Topologie ist eigentlich eine Erweiterung der Linien-Topologie, d.h. das erste und das letzte Gerät sind miteinander verbunden. Ein wesentliches Merkmal dieser Topologie-Art ist, dass jede Station einen eindeutigen Vorgänger und einen deutlichen Nachfolger hat, spricht man daher von einer deterministischen Netzwerkkommunikation. Abbildung 4 zeigt eine Ring-Topologie im Einsatz, dieses System ist sehr einfach aufzubauen.

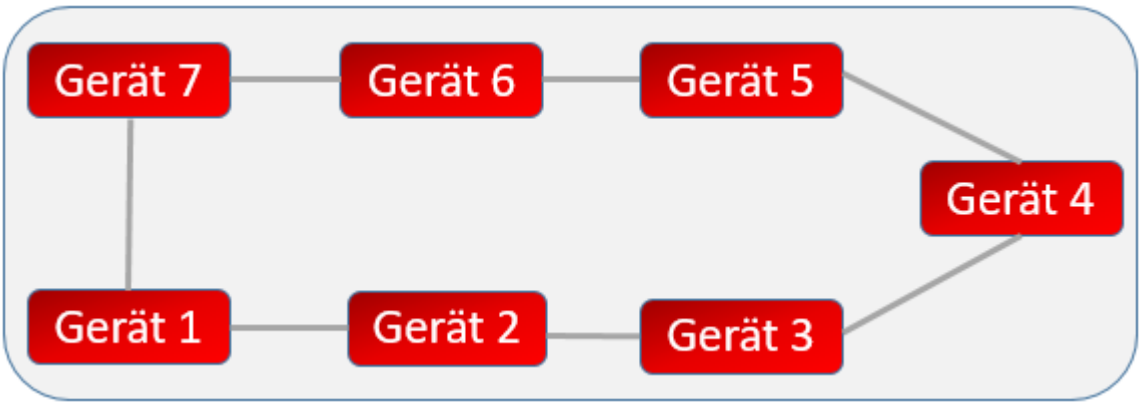


Abbildung 4: Ring-Topologie

Vorteile

- Das System gewährleistet, dass Alle Stationen im gesamten Netzwerkstrang gleiche Zugriffsmöglichkeiten haben
- Da das System eine schnellere Datenübertragung bietet, wird die Übertragungsbandbreite[9] garantiert.
- Jeder Station arbeitet als Verstärker, d.h. eingehende Daten werden verstärkt und weiterleitet
- Da die Anzahl der Verbindungen pro Station (Grad) bei Erweiterung konstant bleibt, ist das System gut skalierbar.
- Das Netz ist leicht programmierbar, denn die gesamten Stationen der Netzwerkstrukturierung besitzen dieselben Anzahl der Verbindungen pro Station

Nachteile

[8] Datenkollisionen bezeichnet man die unerwünschten Umstände, dass Daten verloren gehen, wenn zwei oder mehrere Stationen gleichzeitig über das gemeinsame Medium senden
[9] Übertragungsbandbreite ist eine Kenngröße, die sich aus in einer Zeiteinheit zu übertragenden Datenmengen ergibt

- Es entstehen hohe Kosten für den Aufbau des Netzwerks, da die Komponenten teuer sind
- Der Durchmesser[10] der Topologie ist relativ hoher.
- Höherer Verkabelungsaufwand, Da jede Station zwei Verbindungen besitzt

2.2.3 Stern-Topologie

Charakteristisches Merkmal der Stern-Topologie (Heinrich Hübscher, 2011) sind kurze Wege, das bedeutet, dass zwischen Sender und Empfänger nur wenige Vermittlungstationen vorhanden sind. Abbildung 5 zeigt eine klassische Stern-Topologie, diese besteht aus einem zentralen Element (Gerät 3) und sechs Geräte, die mit einem Ende-zu-Ende Verbindung (ist eine Verbindungsart bei dem Sender und Empfänger direkt miteinander verbunden sind) angeschlossen sind. Fällt ein Gerät aus, stört es die Kommunikation die übrigen Geräte nicht, solange diese nicht mit dem ausgefallenen Gerät Kommunizieren. Abhängig von der Ausgestaltung des zentralen Elements unterscheidet man zwischen aktiven und passiven Sternsystemen. Bei ersteren übernimmt das zentrale Element die Vermittlungsaufgaben, bei anderen dient es höchstens der Signalregeneration.

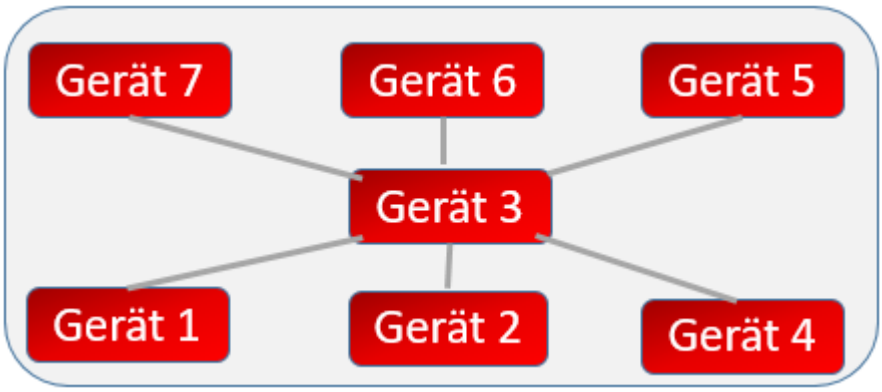


Abbildung 5: Stern-Topologie

Vorteile

- Durch die einfache Verknüpfung der beteiligten Geräte ist das Sternsystem selbsterklärend und leicht verständlich in allen Anwendungsbereichen.
- Das System bietet hohe Übertragungsraten, wenn das zentrale Element ein Switch[11] ist
- Das System ist leicht erweiterbar und ermöglicht einen sehr flexiblen Betrieb.
- Die Struktur des Systems ermöglicht eine schnelle Fehlersuche hinsichtlich einzelner Geräte und deren Verbindungen.
- Sehr gute Eignung für Broadcast[12]-Anwendungen.
- Es gibt kein Routine[13] von Daten, sondern die Daten werden bearbeitet und entsperrenden Geräte weitergeleitet.

Nachteile

- Durch Ausfall des zentralen Elementes wird Netzverkehr unmöglich.
- Hohe Kosten entstehen, wenn das zentrale Element leistungsfähig sein muss.

[10] Durchmesser versteht man die maximale direkte Entfernung, die zwischen zwei Stationen besteht
[11] Switch ist ein elektronisches Gerät zur Verbindung mehrerer Stationen in einem lokalen Netzwerk (LAN).
[12] Broadcast ist eine Nachricht, die von einer spezifischen Station zu allen Teilnehmern gleichzeitig übertragen wird.
[13] Routine bezeichnet man als Vorgang, der den Weg zur nächsten Station von Daten bestimmt

- Niedrige Übertragungsrate bei vielen Geräten.
- Wenn das zentrale Element bereits am Limit arbeitet, so wird es eindeutig zu einer steigenden Verzögerung und damit zu einer niedrigeren gefühlten Performance führen.
- Die Skalierbarkeit des Netzwerkes verursacht häufig einen hohen Zeit- und Kostenaufwand.

2.2.4 Baum-Topologie

Die Baum-Topologie (Conrads, 2014) ist eine erweitere Stern-Topologie, Abbildung 6 zeigt eine solche Struktur an, Gerät 1 repräsentiert das Wurzel, Gerät 2 und 3 stellen die Switches dar und andere Geräte die Endstationen. Derzeit ist dies die meist genutzte Netzwerktopologie. Prinzipiell wäre eine Verkabelung von jedem Switch zu jedem andern Switch möglich. Aus der Bezeichnung lässt sich dieses Topologiemsuter mit einem Baum vergleichen. Sie besitzt eine Wurzel und viele Äste.

Diese Topologie wird sehr oft bei großen Gebäuden eingesetzt, weil mehrere Topologien dort miteinander kombiniert werden.

Vorteile

- Wenn eine Endstation in Netz ausfällt, ergibt sich daraus keine Konsequenzen für die restlichen Geräte.
- Durch die Anordnung der Stationen lässt sich das gesamte Netz strukturell leicht erweitern.
- Das Baumsystem gewährleistet eine gute Eignung für spezifische Such- und Sortieralgorithmen.
- Das System ermöglicht der Transport großer Datenmengen über großen Entfernungen und somit spart hohe Kosten.

Nachteile

- Das System erlaubt in Abhängigkeit von der Übertragungsrate nur eine begrenzte Leitungslänge.
- Eine aufwändige Verkabelung entfällt, da die notwendigen Kabel serienmäßig im Gebäudeautomationssystem vorhanden sind.
- Der Ausfall des Wurzelelements hat hohe Auswirkungen auf die Verfügbarkeiten der restlichen Geräte.

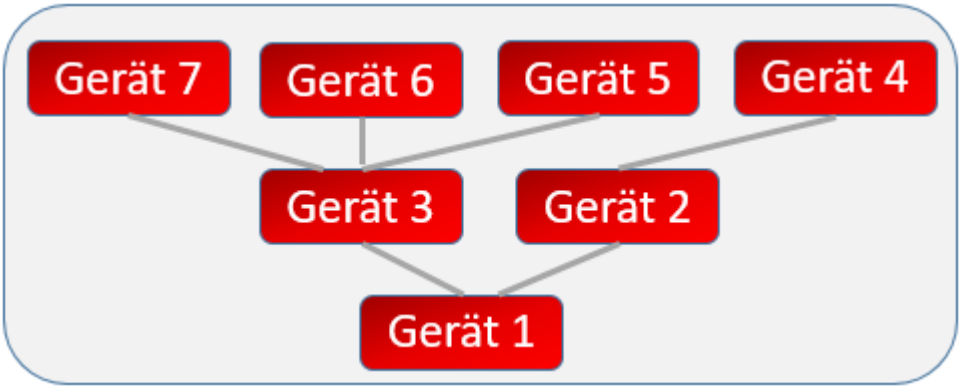


Abbildung 6: Baum-Topologie

2.2.5 Vermaschtes Netz

Es gibt zwei Arten von vermaschten: Die erste ist eine vollvermaschte Topologie, wo jede Station mindestens zwei Verbindungen zu einer benachbarten Station hat und die andere ist eine teilvermaschte Topologie wo lediglich einige Stationen über eine Verbindung angeschlossen sind. Abbildung 7 stellet eine teilvermaschte Struktur mit sieben Geräten dar.

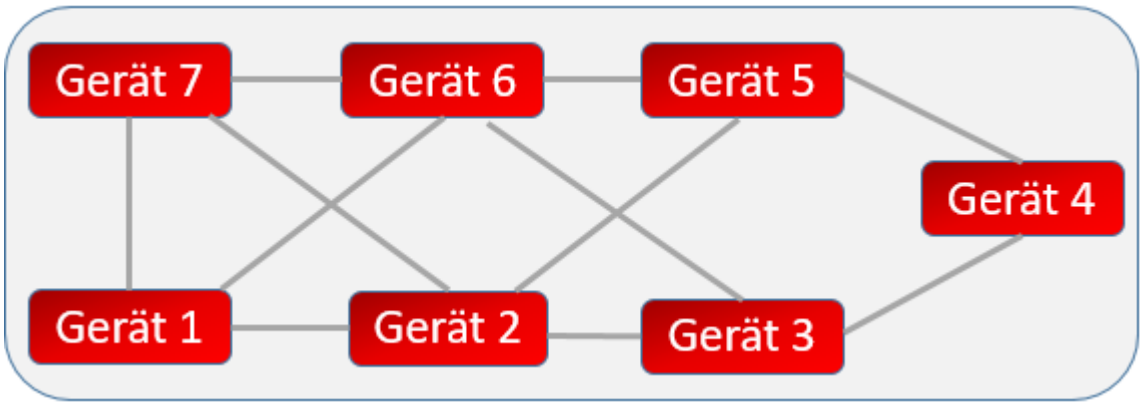


Abbildung 7: Vermaschtes Netzwerk

Vorteile

- Durch die Anordnung der gesamten Struktur ist das System die sicherste Variante eines Netzwerkes
- Bei Ausfall einer Endstation ist durch Umleitung die Datenkommunikation weiterhin möglich, wenn die vollvermaschte Topologie eingesetzt wird.
- Beide Topologie-Arte sind äußerst leistungsfähig und bieten zuverlässige Sicherheit bei gleichzeitigem Bedienkomfort für die Verwaltung eines Netzwerkes.

Nachteile

- Dadurch alle Stationen miteinander im Netz verbunden sind, ist der Verdrahtungsaufwand sehr hoch
- Das System erfordert einen sehr hohen Energieverbrauch und entsprechenden steigende Energiekosten.
- Da das System zuverlässig aufgebaut ist, benötigt er vergleichsweise komplexes Routing

2.3 Netzwerke-Protokolle

In der industriellen Kommunikation gibt es viele verschiedene Netzwerkprotokolle, die zum Einsatz kommen, beispielsweise PROFIBUS, CANopen, DeviceNet, Sercos, EthetCAT, PROFINET. In diesem Abschnitt werden zwei der wichtigsten Netzwerkeprotokolle kurz betrachtet. Bei der Vorstellung handelt es sich um die Protokolle PROFIBUS und PROFINET, da auch der Schwerpunkt dieser Arbeit auf diesen beiden Systemen beruht.
In dem Abschnitt 2.3.1 wird kurz erklärt was der PROFIBUS kann und kurz auf charakteristischen Merkmale der PROFIBUS-Geräte eingegangen. Das Protokoll PROFINET wird dann im anschließenden Abschnitt 2.3.2 betrachtet.

2.3.1 PROFIBUS

PROFIBUS war einst ein von SIEMENS (AG, 2018) entwickelter Standard für die Feldbuskommunikation in der Automatisierungstechnik. Es ist heute der universelle Feldbus, der breite Anwendung in der industriellen Kommunikation und Gebäudeautomatisierung findet. Er ermöglicht die Kopplung von Geräten verschiedenen Hersteller ohne besondere Schnittellenanpassung.

PROFIBUS in das OSI-Modell

Die Protokollarchitektur ist keine Architektur, welche für diese Arbeit die entscheidende Rolle spielt, doch verdeutlicht es Informationen, die in den folgenden Kapiteln verwendet werden. Die PROFIBUS-Architektur ist basiert auf das OSI-Referenzmodell(siehe Abschnitt 2.1) und unterteilt den Ablauf einer Kommunikation in drei Schichten, die im nachfolgenden Abbildung fett dargestellt sind.

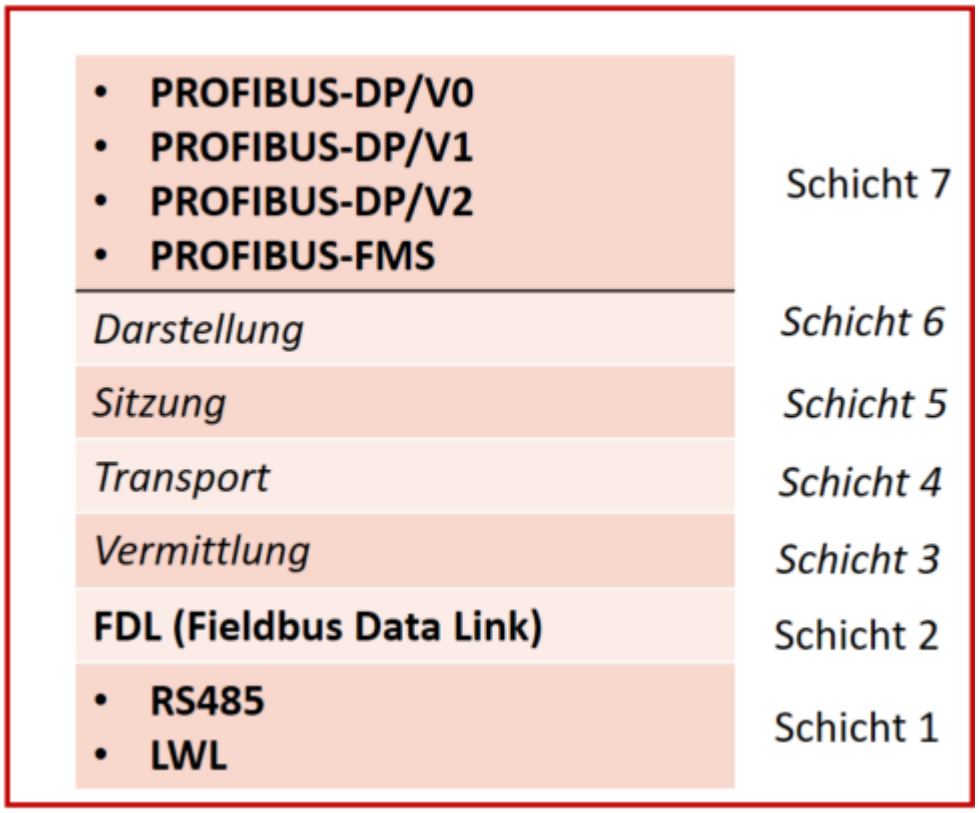


Abbildung 8: PROFIBUS basiert auf international etablierte Standard

Jede Schicht übernimmt genaue festgelegte Aufgaben. Die Schichten 1 bis 6 basieren ausschließlich auf das ISO/OSI-Modell und die Schicht 1 definiert die Funktionalität von PROFIBUS.

PROFIBUS Gerätetypen:

- **Master** – er ist typischerweise eine Steuerung, die den gesamten Datenverkehr initiiert. Er greift auf Resources zu und versendet Nachrichten ohne externe Aufforderungen.
- **Slave –** er ist in einem Feldbussystem ein passiver Kommunikationsteilnehmer, der von dem Master gesteuert wird. Er muss für einen Buszugriff direkt von dem Master aufgefordert werden.

PROFIBUS Varianten:

- **PROFIBUS-DP** (Dezentrale Peripherie): dient zur Kommunikation zwischen zentralen Automatisierungsgeräten und dezentralen Feldgeräten, durch eine serielle Verbindung. Technische Weiterentwicklungen sind der PROFIBUS-DP/V1 und der PROFIBUS-DP/V2.
- **PROFIBUS-PA** (Prozess-Automation): welcher vorrangig im Gebiet der Verfahrenstechnik eingesetzt wird, ist für die Kommunikation zwischen Mess- und Prozessgeräten oder zwischen Aktoren und Prozessleitsystemen zuständig.
- **PROFIBUS-FMS** (Fieldbus Message Specification): heute wird diese Variante nach und nach von PROFIBUS DP übernommen und war in der Automatisierungstechnik für komplexe Maschinen und Anlagen vorgesehen.

Die charakteristischen Merkmale eines PROFIBUS-Gerätes werden in Form eines elektronischen Gerätedatenblattes oder Gerätebeschreibungsdatei, der sogenannten GSD-Datei, beschrieben und festgelegt. Die Integration eines Gerätes findet anhand einer solchen Gerätebeschreibungsdatei statt.

Mit dem PROFIBUS können folgende System Systeme realisiert werden:

- Master-Slave-System
- Master-Master System
- Master/Slave–Master/Slave Mischsystem

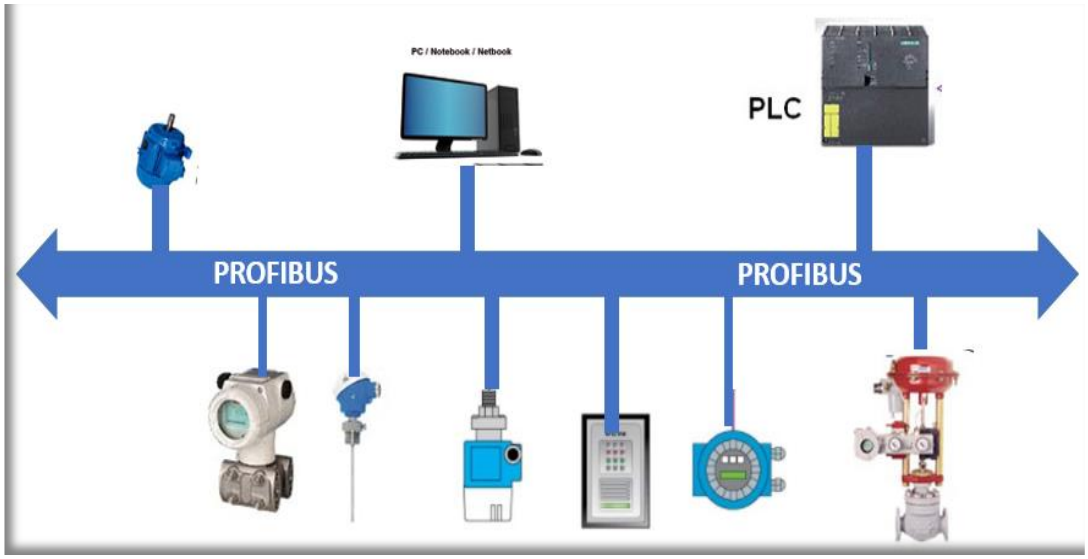


Abbildung 9: Beispielhafte PROFIBUS-Topologie an einem Master-Slave-System

Abbildung 9 zeigt ein PROBUS PA-Topologie auf Basis eines Master-Slave-Systems. Das System besteht aus einer Master-Station (SPS/PLC) den angeschlossenen Slave-Stationen. In diesem System können zyklischen als auch azyklische Daten zwischen dem Master und den Slaves ausgetauscht werden. Die SPS zur Steuerung des gesamten Systems und der PC zur Konfiguration des Systems und zur Programmierung der SPS. Der kommuniziert in der Praxis meist über ein eigenes Protokoll mit der Steuerung.

2.3.2 PROFINET

PROFINET wurde zusammen von Siemens und den Mitgliedsfirmen der Profibus-Nutzerorganisation entwickelt. Basierend auf Ethernet-TCP/IP ist PROFINET die Ergänzung zur PROFIBUS. Durch das Ethernet-TCP/IP besitzen Anwendungen nun die Möglichkeit, neben der schnellen Datenkommunikation IT-Funktionalitäten durchzuführen. Das PROFINET-Netzwerk wird wie auch PROFIBUS in der Prozessautomatisierung und Gebäudeautomation eingesetzt.

PROFINET Gerätetypen:

- **Controller:** ist eine Steuerung, die das System steuert.
- **Device:** ist ein Gerät, welches von einem Controller gesteuert wird. Ein Device besteht aus Modulen und Submodulen.

PROFINET Varianten:

- **PROFINET IO** (Input/Output): ist für die Anbindung an eine Steuerung gedacht und kann als direkter Nachfolger für PROFIBUS-DP gesehen werden
- **PROFINET CBA** (Component Based Automation): ist eine komponentenbasierte Kommunikation im modularen Anlagenbau gedacht.
- **PROFINET IRT** (Isochronous-Real-Time): ist der taktsynchrone Datenaustausch im PROFINET. Die Datenaustausch-Zyklen liegen normalerweise im Bereich von wenigen hundert Mikrosekunden bis zu einer Millisekunde. Der Unterschied zur Real-Time-Kommunikation liegt im Wesentlichen im Determinismus.

Auch beim PROFINET sind die Funktionalitäten und Eigenschaften in einer Gerätebeschreibungsdatei beschrieben. Diese Datei enthält neben den relevanten Daten für das Engineering[14] auch Daten, die für den Datenaustausch mit dem Device von Bedeutung sind. Die Nutzerorganisation schreibt zwingend die Lieferung einer GSDML-Datei vor. Diese ist Bestandteil des Zertifizierungstests.

[14] widmet sich der technischen Forschung und Entwicklung, sowie der Produktionstechnik in der industriellen Kommunikation auf naturwissenschaftlicher Grundlage

3 Ausgangssituation

Die Firma Hischler hat die Fachkompetenz für die Entwicklung und Produktion industrieller Kommunikationslösungen für die moderne Fabrikautomation. Mit der Tendenz „Industrie 4.0“ (Heinrich Hippenmeyer, 2016) und den steigenden Anforderungen wurde bei der Firma Hilscher eine neue Software für die Gerätekonfiguration und der Konfiguration industrieller Netzwerke konzipiert. Mit dieser Software Communication Studio soll in Zukunft auch eine webbasierte Konfiguration möglich sein. Durch den modularen Aufbau der Software ist es möglich, den Topology-Editor als Web-Topology-Editor zu konzipieren.

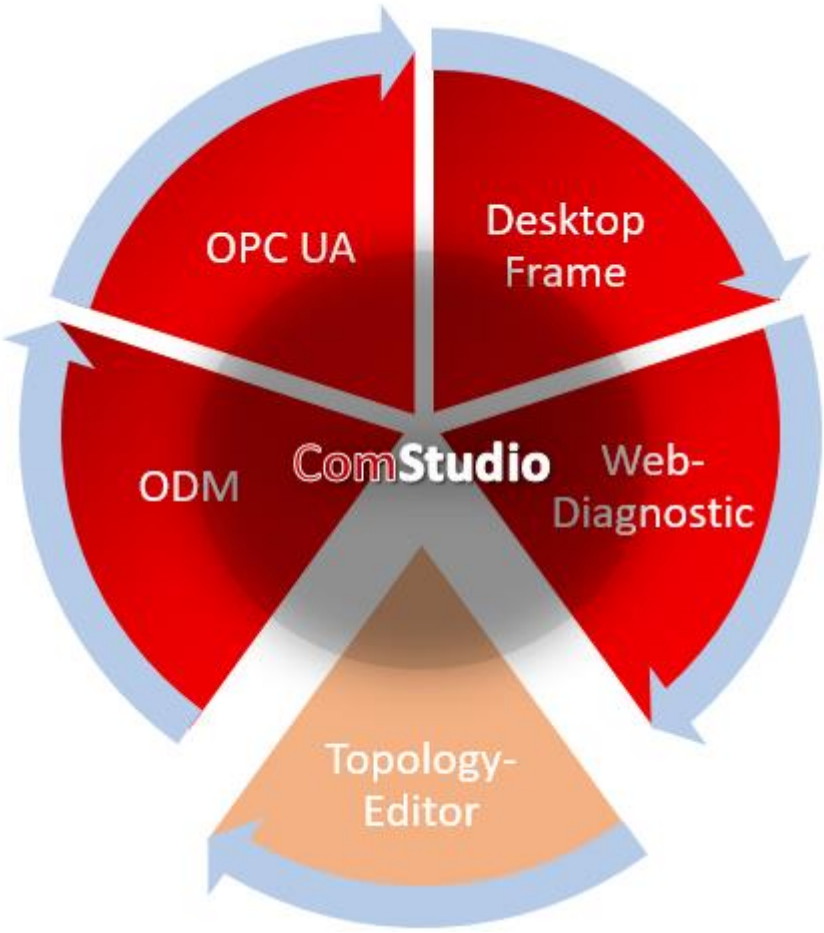
Die Ist-Aufnahme von Communication Studio wurde nach den im Grundlagenkapitel vorgestellten Anforderungen erarbeitet. Für die Erfassung der Topologie-Informationen hat sich eine Vielzahl von Methoden herausgebildet. Hierbei werden zunächst die verwendeten Befragungsmethoden für die Ist-Zustand-Analyse kurz vorgestellt.

3.1 FDT

FDT ist eine Technologie, die den Datenaustausch zwischen Feldgeräten und Automatisierungssystemen unterstützt. Die Technologie basiert auf einer Schnittstellenspezifikation, die als IEC 62453 standardisiert ist. Die Spezifikation definiert zwei Hauptkonzepte: Device Type Manager (DTM) und Frame Applikation. Ein DTM ist eine Softwarekomponente, die für einen Feldgerätetyp spezifisch ist. Eine Rahmenanwendung ist eine Softwareumgebung (Teil des Automatisierungssystems) zur Integration von DTMs. Innerhalb einer Rahmenapplikation stellt jeder DTM spezifische Daten und Dienste für das jeweilige Feldgerät zur Verfügung. Da die Technologie auf einer standardisierten Schnittstelle basiert, kann jeder DTM in jede Rahmenanwendung integriert werden. Basierend auf FDT ist es möglich, Kommunikationsgeräte, Kommunikationsinfrastrukturgeräte (z. B. Gateways) und Feldgeräte abhängig von ihren Kommunikationsprotokollen zu integrieren. Die Unterstützung verschiedener Kommunikationsprotokolle wird durch zusätzliche Kommunikationsprotokollspezifikationen z. B. für PROFINET und PROFIBUS (siehe Abschnitt 3.4), Ethernet IP, DeviceNET, HART und CANopen oder durch herstellerspezifische Protokollintegration wie Beispielerweise Hilscher-Protokoll bereitgestellt. Die aktuelle Version ist FDT2.0.

3.2 Was ist Communication Studio?

Communication Studio, in dieser Arbeit abgekürzt auch ComStudio genannt, ist der Name für das neue Softwareprodukt der Firma Hilscher. Diese Software soll zukünftig die bisherige Software SYCON.net ablösen und ist vollständig auf FDT2.0 basiert. Grund für die Ablösung der Software durch das ComStudio sind unter anderem die schlechte Erweiterbarkeit und dadurch in Zukunft fehlende Funktionalität.
Das ComStudio ist modular aufgebaut und besitzt die Möglichkeit, Softwaremodule über eine Plugin Schnittstelle zu integrieren. Vorrangig wurde für die Programmierung C#[15] verwendet und die Oberflächen wurden mit WPF[16] erstellt.
Durch geringfügige Anpassungen konnten einige bestehenden Konfigurationsmodule aus der Software SYCON.net im ComStudio wiederverwendet werden. Andere Teile, wie z.B. die Diagnose der Baugruppen wurden neu entwickelt. Dieser Programmteil wurde als „*Web-Diagnostic“* als Client für browserbasierte Anwendungen auf HTML-, Angular- oder TypeScript-Basis erstellt. Ein Grund hierfür ist die häufigere Nutzung ohne Änderung der Konfiguration und somit ist ein Start der gesamten Software nicht nötig.
Abbildung 10 zeigt die wichtigsten Module des „Communication Studios“. Diese Module bieten den Vorteil, dass die Software mit den steigenden Anforderungen mitwachsen kann.



[15] ist eine objektorientierte Programmiersprache von der Firma Microsoft und ist plattformunabhängig.
[16] ist eine Klassenbibliothek zur Entwicklung von grafischen Benutzeroberflächen.

Abbildung 10: Vereinfachtes Übersichtschema ComStudio mit zukünftiger Topology-Editor

3.3 Befragungsmethoden

Damit eine umfassende Darstellung des Ist-Zustandes hinsichtlich der Visualisierung der Geräte gewährleistet werden kann, ist es von entscheidender Bedeutung, zuerst eine gute Informationsgrundlage zu schaffen. Eine der wichtigsten Informationsquellen sind hierbei die Mitarbeiter der Firma Hilscher, da sie die Arbeitsabläufe bei der Inbetriebnahme kennen und über Erfahrungen mit den Spezifikationen verfügen.
Folgende Methoden werden genutzt, um die Informationen zu erfassen:

Regelmäßige Meetings
Bei dieser Methode werden die direkten mündlichen Fragen gestellt. Die Befragung wird entweder auf Basis eines Fragenkatalogs durchgeführt oder Fragen, die sich im Verlauf des Meetings ergeben.

Dokumentenanalyse
Bei dieser Analyse werden Informationen aus Spezifikationen ermittelt und die Anforderungen der bisherigeren Softwareanwendung ComStudio extrahiert.

Beobachtungen
Durch Beobachtung werden die Sachverhalte durch sinnliche Wahrnehmung aufgenommen. Ich habe den Mitarbeitern bei der Arbeit und beim Umgang mit dem ComStudio zugesehen. Aus diesen Beobachtungen habe ich viele Information gewonnen wie z.B. die Reihenfolge der Bedienung / Fenster oder populärsten Parameter.

3.4 Auswertung

Diese Auswertung des Ist-Zustandes und die Befragung der Mitarbeiter dienen als Grundlage für den Soll-Zustand.

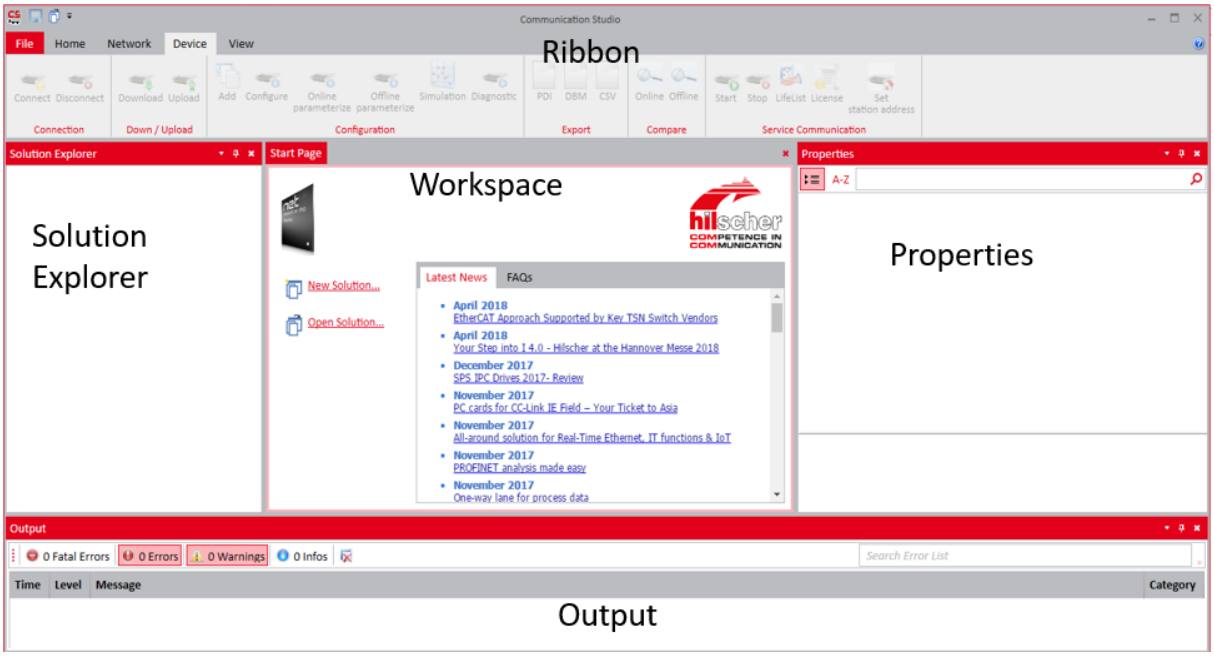


Abbildung 11: Benutzeroberfläche von ComStudio mit spezifische Ansichten

Abbildung 11 zeigt die Standard-Ansichten der bestehenden Software, hier sieht der Nutzer die Liste der Projekte welche in eine Solution zusammengefasst sind, detaillierte Information spezifische Gerätetype bzw. Verbindungstypen, Ergebnisse in bereits geschlossen Prozesse und kann in Workspace eine neue GUI öffnen.

Im Folgenden werden nur die wichtigsten Punkte, die im Zusammenhang mit der Topologie stehen, gelistet:

- Das bisherige System bietet nur eine hierarchische Darstellung der Geräte im sogenannten „*Solution Explorer*“ (Abbildung 11) an. Allerdings ist diese Art der Darstellung kein Ersatz für eine topologische Darstellung eines Netzwerkes.
- Die Darstellung, Bearbeitung und das Handling der Topologie soll einheitlich für alle Netzwerke sein.
- Communication Studio ist nicht cloud-fähig. Die Firma Hilscher sieht zukünftig die Möglichkeit des Einsatzes von Cloud-Konfiguratoren und setzt daher auf neue Technologie.
- Modularer Aufbau mit Plugin Schnittstelle
- Topologie wird zur Konfigurations- und Diagnosezeit benötigt.

- Das System bietet schon eine Möglichkeit die detaillierte Information über die Kommunikationsgeräte, die Kommunikationsinfrastrukturgeräte und Verbindungen.in einer sogenannte „*Properties*“-Fernster. (Abbildung 11)
- Das System bietet eine Multifunktionsleiste(„*Ribbon*“), die Elemente Menüsteuerung, Symbolleisten und Dialoge miteinander verbindet, sie wiederspiegelt ein grafisches Bedienkonzept für das Anwendungsprogramm.

4 Anforderungsanalyse

Um den Aufwand für Planung und Implementierung einschätzen zu können, ist eine genaue Analyse der verschiedenen Anforderungen an das System notwendig. Die Analyse muss umfassend und detailliert genug sein, sodass es keine Missverständnisse oder Interpretationsmöglichkeiten während der Entwicklung gibt.
Eine gute Analyse ist der Grundstein für ein realistisches Zeitmanagement und in Projekten für eine realistische Kosteneinschätzung (Sommerville, 2007).
Die folgenden Punkte listen einige Vorteile, warum wir überhaupt diese Analyse grundsätzlich durchgeführt haben:

- Kosten einschätzen / senken
- Risiken verkleinern
- Erleichterung beim Erstellen von Testfällen und Testszenarien
- Gute Produkte erleichtern den Verkauf und Kundenzufriedenheit

Dieser Abschnitt beschreit deshalb alle Anforderungen an den Topology-Editor und das gesamte System. Hierbei wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden. Des Weiteren werden die Aufgaben beispielhaft an einigen Anwendungsfällen in einem Diagramm erläutert.
.

4.1 Funktionale Anforderungen

Die funktionalen Anforderungen (FR[17]) spezifizieren die Funktionalität oder das Verhalten des Softwareproduktes unter festgelegten Bedingungen oder als Frage „Was soll das Softwareprodukt tun?“.
Hierbei gibt es die Muss-Kriterien, das heißt die Kriterien, die auf jeden Fall umgesetzt werden müssen. Sie beschreiben vor allem den Einsatz essentieller Funktionen des Topology-Editors. Weiterhin formuliert man die Kann-Kriterien, die umgesetzt werden, wenn es die Ressourcen und die Planung des Projektes zulassen und nicht die Einsatzfähigkeiten des Systems beeinträchtigen.

4.1.1 Muss-Kriterien

FR010: Die Topologie ist in einer Ansicht darzustellen.

FR020: Die Topologie-Darstellung kann separat geschlossen werden.

[17] Functional Requirement

FR030: Die Topologie-Darstellung eines Projekts kann durch den Anwender separat geschlossen werden.

FR040: Das Editieren von Informationen einer Topologie durch den Nutzer muss möglich sein.

FR050: Dem Nutzer muss die Möglichkeiten geboten werden die Topologie wiederzuverwenden.

FR060: Nach dem Öffnen sind die wichtigsten Informationen in einer Ansicht bereitzustellen.

FR070: Die topologischen Informationen müssen grafisch dargestellt werden.

FR080: Den Nutzer beim Editieren einer Verbindung optisch unterstützen.

FR090 Die Basisfunktionen wie Laden, Speichern, Importieren und Exportieren müssen bereitgestellt werden.

FR100 Darstellung verschiedener Topologien.

FR110 Die Topologie-Ansicht zoomen.
.

4.1.2 Kann-Kriterien

FR200: Bereitstellen einer „Über Uns“-Ansicht.

FR210: Funktion zum Umstellen der Sprache.

FR220: Funktion zur Umschaltung der Topologie-Art (siehe Abschnitt 2.2)

FR230 Funktion zum Suchen und Ersetzen von Informationen.

FR240 Funktion zur Filterung von Informationen und der Darstellung.

FR250 Bereitstellen der Topologie unabhängig vom Protokoll.

FR260 Darstellen von Fehlern in einer eigenen Ausgabe

FR270 Bereitstellen einer Auto-Sortierungs-Funktion für die Darstellung.
.

4.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen (NFRs[18]), auch technische Anforderungen genannt, beschreiben Aspekte, die typischerweise mehrere oder alle funktionalen Anforderungen betreffen bzw. überschneiden (cross-cut). Sie haben in der Regel einen Einfluss auf die gesamte Softwarearchitektur. Außerdem beeinflussen sich nichtfunktionale Anforderungen gegenseitig. Auch hier werden die Anforderungen in Muss-und in Kann-Kriterien aufgeteilt.

4.2.1 Muss-Kriterien

NF010: Es muss möglich sein, die Software zu erweitern oder Fehlerkorrekturen durchzuführen (Wartbarkeit).

NF020: Die Software muss in der Lage sein, Mängel oder Ursachen von Fehlverhalten zu diagnostizieren oder änderungsbedürftige Teile zu identifizieren (Analysierbarkeit).

NF030: Die Software muss die Änderung der Implementierungen einer Spezifikation zulassen (Änderbarkeit).

NF040 Die Software soll fehlertolerant sein (Stabilität).

NF050: Die Software soll validierbar sein (Testbarkeit).

NF060: Die Software ist ein Bestandteil des Communication Studio.

NF070: Der Datenaustausch erfolgt im XML-Format.

NF080: Die Software muss unter Webbrowsern Google Chrome, Mozilla Firefox und Internet Explorer benutzbar sein.

.

4.2.2 Kann-Kriterien

NF110: Die Software soll zukünftig in einer Cloud zur Verfügung gestellt werden.

NF120: Die Software soll mit Angular 5 umgesetzt werden.

4.3 Anwendungsfälle

Um einen Einblick in den häufigsten genutzten Funktionsumfang geben zu können, soll daher nachfolgend eine Auswahl von Anwendungsfällen dargestellt und erörtert werden. Abbildung 12 gibt bereits einen Überblick über einige der Anforderungen.
Prinzipiell lassen sich bei der Interaktion mit dem System zwei Nutzerrollen identifizieren. Bei Anzeigen und Erstellen der Netzwerktopologie sind hier der Kunde und Inbetriebnehmer zu nennen.

[18] Nonfunctional Requirement

Der Kunde bzw. Inbetriebnehmer der Firma Hilscher möchten im System eine Topologie anzeigen. Sie sollen durch Communication Studio das entsprechende Projekt anlegen oder importiert (falls schon vorhanden ist) die Geräteladen. Nach erfolgreicher Einfügen der DTMs, kann der Kunde die Topologie anzeigen. Somit können Sie hier ohne Rechte die entsprechende Netzwerktopologie bearbeiten.

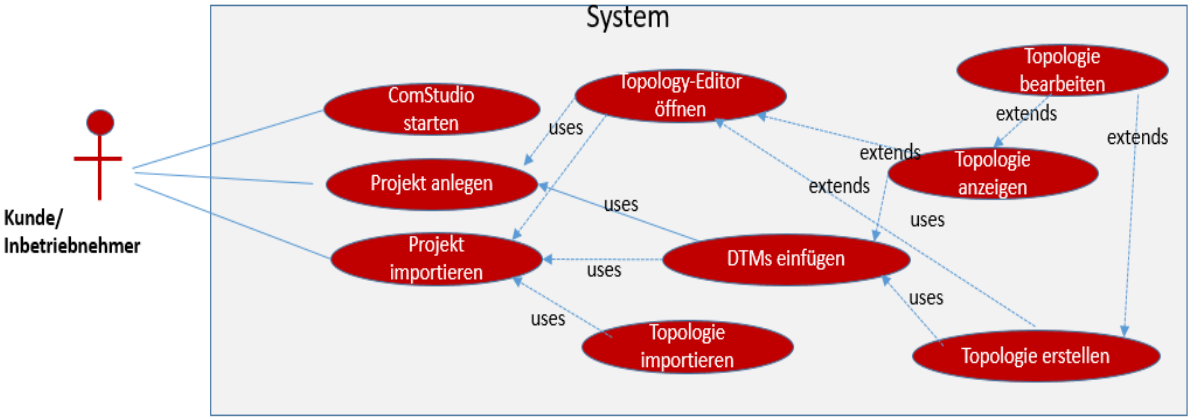


Abbildung 12: Anwendungsfall-Diagramm: Ausgewählte Funktionen des Systems

Während der Bearbeitung haben die Anwender prinzipiell zwei Optionen:

- **Hinzuzufügen eines neuen DTM**: Communication Studio bietet schon ein Verfahren um ein DTM in das System hinzufügen. Der Nutzer sollte einfach die Funktion aufrufen. Die Topologie wird direkt mit dem neuem DTM aktualisiert.
- **Löschen eines bestehenden DTM**: Um ein DTM zu löschen soll der Nutzer zu allererst das DTM aus dem Topology-Editor selektieren, dadurch wird die DELETE-Funktion aktiviert, die im Ribbon von Communication Studio liegt.

5 Evaluation der Technologien

Topology-Editor basiert auf einem verteilten System, das Module zur Umgebungserfassung mit Datenverarbeitungsknoten und Schnittstellen zu andere Programme kombiniert. Der Entwicklungsprozess der Software für ein solches System umfasst neben der eigentlichen Anwendungsentwicklung zwei periphere Aufgaben: Zum einen muss die Kommunikation zwischen Komponenten des Systems sichergestellt und zum andere die Interfaces zur Hardware oder anderen externen Programmen bereitgestellt werden.
Um den Entwicklungsaufwand an dieser Stelle zu reduzieren, muss die in Kapitel 4 beschriebene Vielfalt gekapselt und der Zugriff auf eine uniforme Schnittstelle abgebildet werden. Erst damit ist eine Austauschbarkeit und Wiederverwendbarkeit von Komponenten umsetzbar. Dadurch kann eines der wichtigsten Muss-Kriterien aus der Anforderungen abgedeckt werden. Nach monatelang wurde eine Fülle von verschiedenen Frameworks für die Entwicklung solcher Plugin gefunden. Nunmehr stehe ich aber vor den Aufgaben aus der Vielzahl an verfügbaren Implementierungen, das für meine konkrete Anwendung geeignetste zu finden. Entsprechend muss ich als Entwickler nach dem Abstecken das Szenarienkontexts, wie zum Beispiel der beteiligten Komponenten, den nötigen Algorithmen, den erforderlichen Kommunikationseigenschaften usw., aufwendige Recherche betreiben, um eine weitegehende Übereinstimmung sicherzustellen. Dieses Kapitel dient insbesondere im Clientseitige[19] Umsetzung mit vergleichbare Aufstellung als Überblick über den gefundenen Frameworks. Für die serverseitige[20] Umsetzung wird die ASP.NET.Core 2.0 eingesetzt.
Anschließend führen wir das Tools bzw. die externen Bibliotheken zur Vereinfachung und Beschleunigung der Entwicklung. Für die Visualisierung von Information Daten werden die Bibliotheken GoJS (Northwoods Software, 1998), KendoUI (Progress Software Corporation , 2018) und SAPUI5 (SAP, 2018) evaluiert.
Das Styling und die Strukturierung von Benutzeroberfläche wird das Bootstrap (Twitter, 2010) verwendet.

5.1 Clientseitige relevante Frameworks

In den letzten Jahren hat sich im Bereich der Webtechnologie eine Entwicklung von browserfähigen Anwendungen, die für die Erfüllung spezifischer Aufgaben als eine kompakte Einheit entworfen wurden, hinzu autonomen, modularen Systemen vollzogen. Kennzeichnend für diese neue Webtechnologie-Generation ist die Kooperation und Interaktion zwischen Software-Modulen. Um die Integration diese Komponente in einem Gesamtsystem zu ermöglichen und dabei den Software-Entwicklungsprozess zu vereinfachen und zu beschleunigen, sind viele Frameworks mit unterschiedlichen Schwerpunkten entwickelt worden. Diese gehen zum Beispiel

[19] beschreit ein Computerprogramm, das Dienste von einem Server abruft
[20] beschreibt ein Computerprogramm, welches auf einem zentralen Computer ausgeführt wird

Kommunikation und Konfiguration, die mit einem modularen System einhergehen, an. Im Folgenden stützt sich dieser Abschnitt auf die wichtigsten Frameworks, die zum einen eine Erfüllung des in Abschnitt 4.1 und 4.2 beschriebenen Aspekte bieten und zum anderen eine Verbreitung gefunden haben.

5.1.1 JavaScript Frameworks

Angular

Das Entwickeln und Erstellen von HTML-Seiten[21], die flexibel und einfach zu warten sind, kann eine Herausforderung darstellen. In diesem Abschnitt werden einige der häufigsten Probleme beschrieben, wie bei der Erstellung von HTML auftreten können, und es wird beschrieben, wie Angular-Technologie diese Herausforderungen meistern kann.

In der Regel stehen Entwickler von HTML-Seiten vor einigen Herausforderungen. Anwendungsanforderungen können sich im Laufe der Zeit ändern.
Neue Geschäftschancen und –Herausforderungen können sich ergeben, neue Technologien werden möglicherweise verfügbar, oder sogar das kontinuierliche Kundenfeedback während des Entwicklungszyklus kann die Anforderungen der Anwendung erheblich beeinflussen. Daher ist es wichtig, die Anwendung so zu erstellen, dass sie flexibel ist und im Laufe der Zeit leicht geändert oder erweitert werden kann. Entwerfen für diese Art von Flexibilität kann schwer sein zu erreichen denn es erfordert eine Architektur, die es ermöglicht, einzelne Teile der Anwendung unabhängig voneinander zu entwickeln und zu testen, die später isoliert oder aktualisiert werden können, ohne den Rest der Anwendungen zu beinträchtigen

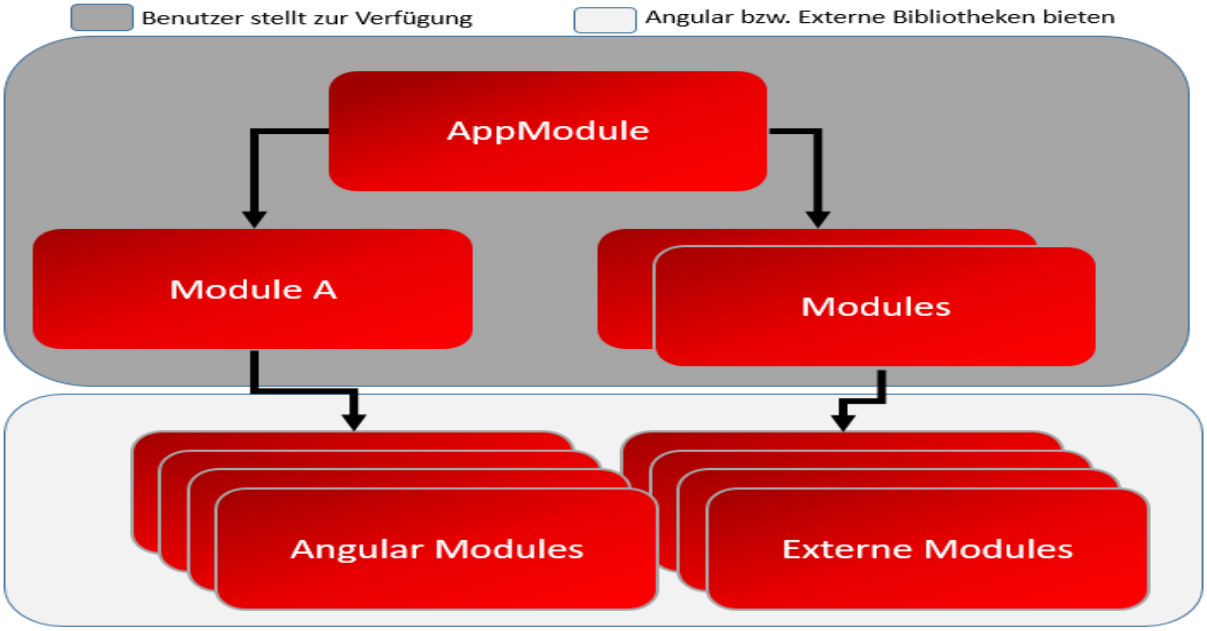


Abbildung 13: Typische zusammengesetzte Anwendungsarchitektur mit dem Angular Framework

Die Wichtigsten Konzepte von Angular

Dieser Abschnitt bietet eine kurze Übersicht über die wichtigsten Konzepte von Angular und definiert einige Begriffe, die in der Dokumentation und in Quellecode verwendet werden

[21] sind Seiten zur Strukturierung digitaler Dokumente mit Inhalten wie Hyperlinks, Bildern usw.

- **Module**:: Module sind Pakete mit Services, Components, Direktives und Templates, die unabhängig voneinander entwickelt, getestet, und optional bereitgestellt werden können. In vielen Situationen werden Module von separaten Teams entwickelt und gewartet. Ein typisches Angular-Projekt besteht aus mehrere Modulen. Sie können verwendet werden, um bestimmte Geschäftsbezogene Funktionen (z.B. Editieren von Topologie, Startseite verwalten) darzustellen und alle Ansichten, Dienste und Datenmodelle zu kapseln, die zum Implementieren dieser Funktionalität erforderlich sind.
- **AppModule**: In einer Modulare Applikation müssen Module zur Laufzeit von der Hostanwendung erkannt und geladen bzw. importiert werden. In Angular wird ein AppModul verwendet, um anzugeben, welche Module importiert werden sollen und in welcher Reihenfolge.
- **Component**: Bei Angular ist keine direkte Manipulation des HTML-DOM (MDN web docs, 2005) vorgesehen, sondern eine Trennung nach MVC-Muster[22]. Der Entwickler definiert dafür ein Tag[23] in einer Seite, der durch eine mit *@Component* dekorierte Komponentenklassen repräsentiert. Der *Selector* legt den Tagnamen fest. Die mit der Komponente verbundene Vorlage (*Template*) kann der Webentwickler direkt als Zeichenkette in der Eigenschaft *template* angeben, was sich aber nur bei sehr kurzen HTML-Blöcken anbietet. Eine Komponente kann auch eigene CSS-Vorlagen (w3schools.com, 1999) (Inline oder als eigenständige Datei) besitzen, die allein für die Darstellung dieser Komponente gelten. Einer der wichtige Verhalten der Komponente ist, dass Komponenten im Laufe Ihrer Lebenszyklen diverse Ereignisse auslösen, in die der Entwickler sich einhängen kann.
- **Template**: Angular Template ist die benutzerorientierte Vorlage, die beschreibt was der Benutzer sehen und tun kann. Die Verwendung von solchem Template zur Erstellung einer Ansicht bietet den Vorteil, dass man nicht jedes Detail der Ansicht erstellen muss. Angular Templates erlauben unterschiedliche Syntaxen wie Interpolation, Template Statement usw. Als eigenständige Datei lässt sich die Angular Template in einer Applikation durch die Dateienden *.html* erkennen.
- **Services**: Angular Services werden häufig Zum Speichern von Daten und zum Ausführen von http-Anfragen verwendet. Er ist eine breite Kategorie, die jeden Wert, jede Funktion oder jede Funktion umfasst, die eine Komponente benötigt. Ein Service ist typischerweise eine Klasse mit einem engen, genau definierten Zweck. Es sollte etwas Bestimmtes tun und es gut machen. Angular bietet Solche Services, um die Modularität und Wiederverwendbarkeit zu erhöhen. Eine Komponente sollte beispielsweise nicht definieren müssen, wie Daten vom Server abgerufen, Benutzereingaben validiert oder direkt in der Konsole protokolliert werden. Stattdessen kann es solche Aufgaben an Dienste delegieren (Google, 2018)

Vue.js

Vue ist ein fortschrittliches Framework für den Aufbau von Benutzeroberflächen, das von Evan You (LinkedIn, 2018) im Jahr 2013 entwickelt wurde. Die Aktuelle Version ist 2.5.16 (Vue.Js, 2018). Es ist von Grund auf so konzipiert, dass es stufenweise adoptierbar ist, und kann je nach Anwendungsfall leicht zwischen einer Bibliothek und einem Framework skalieren. Es besteht aus einer zugänglichen Kernbibliothek, die sich nur auf die Ansichtsebene konzentriert, und

[22] ist ein Architekturmuster zur Trennung von Software in drei Komponenten (Datenmodell, Ansicht und Programmsteuerung)
[23] ist ein HTML-Anweisung in spitzen Klammern

einem Ökosystem von unterstützenden Bibliotheken, welche sie bei der Bewältigung der Komplexität großer SPA[24]-Anwendungen unterstützt. Dieses Framework arbeitet nach dem Entwurfsmuster MVVM und ist stark von Angular inspiriert.

React

React ist ein JavaScript-Framework zur Erstellung von Webanwendungen. Es wird seit 2013 von Facebook unter einer Open Source-Lizenz entwickelt und besitzt eine große Online Community mit vielen helfenden Entwicklern und Unterstützern. Die Aktuelle Version 16.3.2 (Facebook/React, 2018) besitzt eine deutliche Verbesserung der Fehlermeldung.

Es handelt sich um ein clientseitiges Frontend-Framework, das heißt, der Quelltext wird für die Ausführung im Browser geschrieben und dient der Implementierung einer Nutzeroberfläche. Das Hauptprinzip ist dabei die Erweiterung des Vokabulars von HTML, damit dieses um Funktionalitäten erweitert wird, die es ermöglichen, dynamische Webanwendungen zu erstellen.
In dynamischen Webseiten und -anwendungen können beispielsweise Quelltext-Ausschnitte in verschiedene Seiten eingebunden werden, um Redundanzen zu vermeiden. Außerdem ermöglicht eine dynamische Seite die Interaktion des Nutzers mit bestimmten Inhalten. Das Gegenteil sind statische Webseiten, die aus unveränderlichen und zusammenhangslosen HTML-Dokumenten bestehen. Die einzige Möglichkeit der Navigation bildet dort die Nutzung von Hyperlinks.

Mit React können interaktive UIs erstellt werden. Es können für jeden Status in einer Anwendung einfache Ansichten entworfen werden, und React aktualisiert und rendert genau die richtigen Komponenten, wenn sich die Daten ändern (Facebook Inc., 2018). Darüber hinaus kann es auch auf dem Server diese UIs erstellen und in mobile Apps mit *React Native* (Facebook Inc., 2018) ausführen.

5.1.2 UI-Bibliotheken

Nachdem wir die JavaScript-Softwarebibliotheken, die ein Grundgerüst für die Ausgabe von User-Interface-Komponenten von browserfähige Anwendungen untersucht haben, werden wir in den folgenden Abschnitten die externen Bibliotheken zur Visualisierung von Information Daten angehen.

SAPUI5

SAPUI5 ist eine JavaScript-Framework, die von der Firma SAP (SAP, 2018) entwickelt wurde. Es besteht aus einem funktionsreichen Kern und einer sehr großen Anzahl von UI-Steuerelementen, die in einer Handvoll Bibliotheken organisiert sind.
Diese Bibliotheken besitzen vor allen eine effiziente Engine zum Erstellen und Aktualisieren der HTML-Steuerelemente, darüber hinaus unterstützen sie das MVC-Konzept und die deklarative UI[25]-Konstruktion.
Die Steuerelemente reichen von einfachen Button-Steuerelemente bis zu komplexen Steuerelementen, beispielsweise Grid-System und Diagram-Steuerelemente.
Für Entwickler bietet SAPUI5 eine leistungsstarke Konzepte:

- Der SAPUI5 bietet eine solide Grundlage, die die Entwicklungsprozesse vereinfacht, indem viele Aspekte der modernen Entwicklung hinter den Kulissen verwalten werden.

[24] steht für Single-Page Application
[25] Steht für User Interface

- Es biete integrierte Unterstützung für Architekturkonzept wie Zwei-Wege-Datenbindung und Routing
- Es biete die Bindung mit JSON, XML und andere Datenformaten

Nachteile ergeben sich allerdings, wenn der Entwickler vordefinierter Controls ändern möchte, diese zu ändern ist jedoch auch nicht so intuitiv wie von SAP eigentlich gewünscht.
Die Einarbeitung dauert länger und die Dokumentation ist verstreut.

Kendo UI

Das bulgarische Consulting-Unternehmen Telerik (Progress Software Corporation , 2018) bietet die Kendo UI-Framework für das mobile und web Einsatzfeld an. Der Code läuft nicht in einer WebView[26], sondern in einer Runtime, die von Telerik-Ingenieuren zur Interaktion mit dem zugrundeliegenden Betriebssysteme entwickelt wurde.
Daher können Sie auf die Steuerelemente des UI-Stacks der jeweils Plattform zurückgreifen. Kendo UI bietet zudem –zur Reduktion des Portierungsaufwands – die Möglichkeiten, die Bedienschnittstelle in einer XML angelehnten Sprache zu beschreiben. Die Laufzeitumgebung erzeugt daraus eine Widget[27]-Struktur.
Eine gute gemachte Kendo-UI Applikation unterscheidet sich daher nicht wesentlich von einer komplett nativen und lässt sich zudem automatisch zwischen Betriebssystemen portieren.

Vorteile

- Kendo UI liefert ein umfangreiches Framework für die responsive Webentwicklung.
- Kendo UI bietet für viele Komponenten eine gute Basis denn Browserkompabilität und responsives Verhalten sind feste Bestandteile
- Kendo UI enthält unterschiedliche Farbschema, neben Standard-Komponenten
- die ausführliche Dokumentation aller Komponenten mit ihren Optionen ist beeindruckend

Nachteile

- Das Laden der Kendo UI-Bibliothek länger dauert länger
- Kendo UI generiert hauptsächlich Markup über JavaScript. was die Anpassung der Klasse schwer macht

GoJS

GoJS (Northwoods Software, 1998) ist eine JavaScript-Bibliothek, mit der auf einfache Weise interaktive Diagramme in modernen Webbrowsern erstellt werden können. Sie unterstützt grafische Vorlagen und Datenbindung von grafischen Objekteigenschaften zum Modellieren von Daten.
Die Anpassung von Aussehen und Verhalten hängt hauptsächlich von der Einstellung von Eigenschaften ab.
Wir werden in diesem Abschnitt eine kleine Einführung über GoJS bezüglich die Diagramm
Danach die Nachteile Bzw. Vorteile bei Verwendung von dieser externen Bibliothek

Einführung:

Um eine Netzwerktopologie mit Hilfe von GoJS-Diagramm API bereitzustellen, benötigt GoJS ein Modell, das die spezifischen Anwendungsdaten enthält.

[26] ist eine Komponente zur Darstellung von Web-Inhalten, z.B. HTML-Dateien mit JavaScript und CSS
[27] ist eine Komponente eines grafischen Fenstersystems.

Jedes Diagramm besteht aus Knoten, die durch Links verbunden sein könnten und die optional in Gruppen zusammengefasst sein könnten. Alle Diagramm-Teile sind in Schichten zusammengefasst und nach gewünschten Layouts (z.B. Baum-Layout, Linienlayout …) angeordnet.

Das Diagramm-Modell besteht aus 2 wichtigsten Eigenschaften: *nodeDataArray* beinhalten alle Knoten und *linkDataArray* listet alle Verbindungen, die in dem Diagramm dargestellt werden sollen. Abbildung 12 zeigt die Diagrammklasse und ihre Interaktion: das Ansichtsmodell(Model) implementiert Eigenschaften,

an die die Ansicht gebunden werden kann, und benachrichtigt die Ansicht von Statusänderungen über Änderungsbenachrichtigungsereignisse. In der Regel besteht eine Eins-zu-Eins-Beziehung zwischen einer Ansicht(Diagramm) und ihrem Ansichtsmodell(Model)

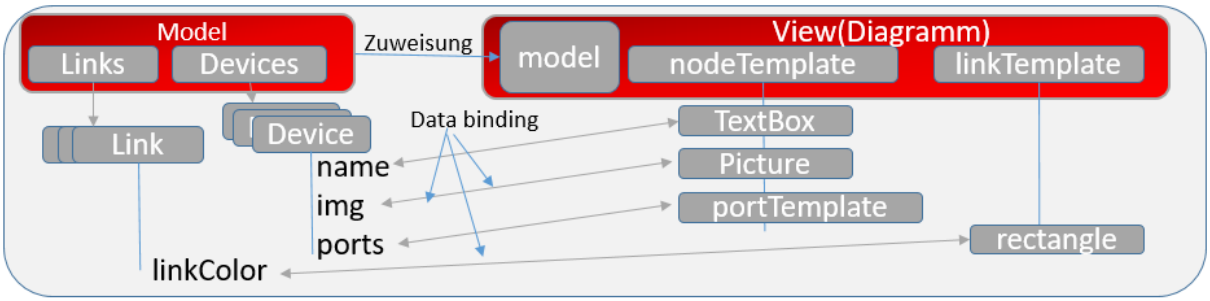


Abbildung 14: Die wichtigsten Eigenschaften des Diagramms und ihre Interaktion

Jeder Knoten oder Link wird normalerweise durch eine Vorlage definiert, die sein Aussehen und Verhalten deklariert. Jede Vorlage besteht aus Gruppen von GraphObjects wie TextBlocks (dient für die Anzeige der Gerätenamen), Picture (dient als Platzhalter für ein Bild) oder Shapes. Es gibt Standardvorlagen für alle Teile, aber fast alle Anwendungen geben benutzerdefinierte Vorlagen an, um das gewünschte Aussehen und Verhalten zu erreichen. Datenbindungen von GraphObject-Eigenschaften zum Modellieren von Dateneigenschaften machen jeden Knoten oder jede Verknüpfung für die Daten eindeutig.

Die Knoten können manuell (interaktiv oder programmgesteuert) positioniert werden oder können automatisch vom Diagramm-Layout angeordnet werden. Knoten besitzen die Möglichkeiten Port als Rechteck darzustellen.

Jedes Diagramm verfügt über eine Reihe von Werkzeugen, die interaktive Aufgaben ausführen, zum Beispiel das Auswählen von Teilen oder das Ziehen von Elementen oder das Zeichnen einer neuen Verknüpfung zwischen zwei Knoten. Der ToolManager [28]bestimmt abhängig von den Mausereignissen und den aktuellen Umständen, welches Werkzeug ausgeführt werden soll.

Jedes Diagramm verfügt außerdem über einen CommandHandler,, der verschiedene Befehle implementiert, z. B. Löschen oder Kopieren. Der CommandHandler interpretiert Tastaturereignisse wie z. B. Steuerelement-Z, wenn der ToolManager ausgeführt wird.

Alle programmatischen Änderungen am Diagramm-, GraphObject-, Modell- oder Modelldatenstatus sollten innerhalb einer einzelnen Transaktion pro Benutzeraktion ausgeführt werden, um sicherzustellen, dass die Aktualisierung korrekt ausgeführt wird und um das Rückgängig-

[28] Das spezielle Werkzeug ist verantwortlich für die Verwaltung aller diagramlosen Werkzeuge

machen / Wiederholen zu unterstützen. Alle vordefinierten Tools und Befehle führen Transaktionen aus, sodass jede Benutzeraktion automatisch rückgängig gemacht werden kann, wenn der UndoManager aktiviert ist. DiagramEvents und Event-Handler auf Diagrams und Graph-Objects sind alle dokumentiert, egal ob sie innerhalb einer Transaktion ausgelöst werden oder ob man eine Transaktion durchführen muss, um das Modell oder das Diagramm zu ändern.

Vorteile und Nachteile von GoJS:
Aus meiner persönlichen Sicht liegen die wesentlichen Vorteile bei dem Einsatz von GoJS-Diagramm in der breiten Unterstützung der Darstellung von Netzwerkstruktur in Webbrowser. Man kann an vielen Stellen ein und dasselbe Problem auf unterschiedliche Art und Weise lösen, ohne sich dabei durch die Vorgaben der Dateninformation eingeengt zu fühlen.
Die Dateiinformation lässt also den Softwareentwicklern den Freiraum, nach eigenen Vorstellungen zu modellieren. Besonders Vorteilhaft ist die Tatsache, dass GoJS auch die Erweiterungsmöglichkeiten durch TypeScript vorsieht und damit wirklich jedem Möglichkeit bietet, den TypeScript-Notationsrahmen zu erschaffen. Diese Möglichkeit sollte aber nur in äußersten Fällen (wenn z.B. keine JavaScript Sprache verwendet werden soll) verwendet werden, da man sonst der Gefahr, sich vom Standard zu entfernen, entgegenläuft.

Die GoJS nutzt die JSON-Notation für die Konfiguration der verschiedenen Klassendiagramme. Diagramm-Konfigurationen können nun nicht nur von den Fachleuten der Software – Firma, sondern auch von den außerhalb stehenden verstanden und verbessert werden.

Weiterer Vorteil liegt in der zunehmenden Verbreitung der GoJS-Bibliothek und sowie dass sie Open-Source ist. Die exakten Zahlen sind zwar noch schwer abzuschätzen, es zeichnet sich jedoch ein wachsender Trend für den Einsatz der GoJS-Bibliothek ab.

Die Nachteile lassen sich nun wiederum aus dem Umfang der Bibliothek und Preise ableiten. GoJS ist sehr vielfältig, so dass auch am Anfang sehr viel Aufwand für das Aneignen und Verstehen sämtlicher Konfigurationen aufgebracht werden muss.

Außerdem wird man in der Regel feststellen, dass viele Elemente sehr selten zum Einsatz kommen und damit eher als Ballast der Bibliothek angesehen werden. Um eine Lizenz zu bekommen, soll man pro Entwickler über 2500 Euro (Northwoods Software, 1998) ausgeben, was zu teuer ist.

5.2 Vergleichskriterien

Nachdem in vorangegangen Abschnitt Beispiele für Clientseitige Frameworks vorgestellt wurden, widmet sich dieser Abschnitt den Kriterien anhand derer ein systematischer Vergleich durchgeführt werden kann. Der Kriterienkatalog integriert dabei Abschnitt zur Lernkurve und Kosten, fasst die Kommunikationsmechanismen zusammen und schließlich werden die Programmierkonzepte betrachtet.

5.2.1 Lernkurve und Kosten

In der industriellen Kommunikation integrieren Konfigurationsapplikationen vielfältige protokollabhängige Feldbusse. Um dennoch Applikationen auf Support-Aufkommen bei sowohl

Nutzer als auch Entwickler reduzieren zu können, spielen Prinzipien wie Kosten und Lernkurve eine wichtige Rolle.

- **Lernkurve**: stellt die Entwicklung eines Trainierenden beim Erlernen eines Frameworks dar, spiegelt somit den Erfolgsgrad des Lernens über den Verlauf der Zeit. Da die Masterarbeit in einer zeitlichen Periode begrenzt ist, sollte diese leicht und schnell wie möglich sein.
- **Kosten**: hier sind die negativen Konsequenzen einer erfolgswirksamen Nutzung von einem Framework. Manche JavaScript-Frameworks sind je nach Zielgruppen relativ teuer. Für ein breites Anwendungsspektrum des jeweiligen Frameworks ist eine Kostenrechnung an die Lizenzbedingungen und Zielgruppennutzung gekoppelt.

5.2.2 Kommunikationsmechanismen

Die Infrastruktur de Kommunikation umfasst Mechanismen, die eine Interaktion und Kooperation zwischen den einzelnen Komponenten des Gesamtsystems ermöglichen und somit dessen Funktionalität als Ganzes gewährleisten.

- **Kommunikationsprinzip** Das Kommunikationsprinzip beschreibt, in welcher Form der Nachrichtenaustauch zwischen Komponenten erfolgt und in welcher Beziehung die Komponenten dabei zueinanderstehen. Die Interaktion kann z.B. über eine Event-Kommunikation und über Prozeduraufrufe erfolgen. Als Interaktionsmuster ist das *Publish-Subscribe*-Modell oder *Producer-Consumer*-Modell denkbar.
- **Kommunikationsmiddleware** Damit Anwendungen verteilt über mehrere Rechnerknoten laufen können und somit eine Ortsunabhängigkeit gewährleisten, sind entsprechende Mechanismen erforderlich.

5.2.3 Programmierkonzept

Diese Kategorie bezieht sich auf Kriterien, die einerseits Eigenschaften der konkreten Implementierung der jeweiligen Entwicklungsumgebung beschreiben und anderseits die Anwendungsentwicklung eben mit dieser betreffen.

- **Unterstützte Programmiersprachen** In der Softwareentwicklung gibt es verschiedene Programmiersprachen, mit denen sich Anweisungen formulieren lassen. Diese Anweisungen können dann von einem Computer ausgeführt werden. Da sich die Anforderung auf einer clientseitigen Anwendung bezieht, sind Programmiersprachen TypeScript bzw. JavaScript denkbar.
- **Unterstützungsbibliotheken** Die externen Bibliotheken liefern vordefinierte Komponenten für spezifische Aufgaben (z.B. Strukturierung und Styling einer Anwendung), wobei gegebenenfalls entsprechende Konstruktionen und Schnittstellen angepasst werden müssen. Diese Bibliotheken erleichtern dadurch die Wartbarkeit und fördern die Wiederverwendung von Softwaremodulen.
- **Lizenzbedingungen** Darin beschrieben sind die rechtlichen Bedingungen für die Verwendung einer Software. Durch das gewählte Lizenzmodell wird, im Falle der Nutzung, die Anwendbarkeit der Software definiert. Bestimmte Lizenzmodelle erleichtern die Entwicklungsarbeit und bieten in Wikis eine Vielzahl von Antworten, Anregungen und Beispielcodes an.

5.3 Ergebnisse der Analyse

Die Entscheidung für und wider clientseitige Framework sollte nicht zwischen Tür und Angel getroffen werden, zu teuer wäre ein Umstieg. Die nachträgliche Erweiterung und Wartbarkeit kommen einem kompletten Rewrite gleich. Tabelle 2 und Tabelle 3 zeigen eine Übersicht der clientseitigen Frameworks bezüglich der jeweiligen Vergleichskriterien:

JavaScript-Frameworks			
	React	**Angular**	**Vue**
Lernkurve	mittel	leicht	mittel
Kostenlose Variante	+(eingeschränkt)	+	+(eingeschränkt)
Kommunikationsprinzip	+	+	+
Kommunikationsmiddleware	+	+	+
Sprachen	TypeScript	TypeScript	JavaScript
Eigene IDE	-	+	-
Open-Source-Framework	+	+	+(teilweise)
Lizenz	+	+	+
Preis	k.A.	k.A.	k.A.
+vorhanden	**-nicht vorhanden**		**k.A keine Angabe**

Tabelle 2: Vergleichstabelle für JavaScript-Framework

UI-Bibliotheken			
	SAPUI5	**GoJS**	**Kendo UI**
Lernkurve	schwer	leicht	schwer
Kostenlose Variante	-	+(einge-schränkt)	+(eingeschränkt)
Kommunikationsprinzip	k.A	+	+
Kommunikationsmiddleware	k.A	+	+
Sprachen	JavaScript	TypeScript	JavaScript
Eigene IDE	-	-	-
Open-Source-Framework	-	-	+(teilweise)
Lizenz	+	+	+
Preis	k.A.	ab 2995 US-$ pro Jahre	ab 899 US-$ pro Jahr
+vorhanden	**-nicht vorhanden**		**k.A keine Angabe**

Tabelle 3: Vergleichstabelle für UI-Bibliotheken besonderes im Fall eines Diagramms

Für die Umsetzung des Topology-Editor wird das Framework Angular in Kombination mit GoJS ausgewählt. Im Vergleich mit den anderen Optionen ist bei diesem Framework an wenigsten geringen plattformspezifischer Code notwendig und der Quellcode kann mit einem vergleichsweise geringen Redundanzen für alle Plattformen verwendet werden. Darüber hinaus ermöglicht es, wartbare Anwendungen zu erstellen. Spezifisches Wissen über dieses Framework ist nur in geringen Maße notwendig, da vorhandenes Wissen aus dem Bereich der Webtechnologie angewendet werden kann, ohne ein Konzept zu erlernen. Die Firma Hilscher hat schon mal das Framework benutzt, um ein Web-Diagnose-Tool (Siehe Abbildung 8) zu entwickeln.

In der Praxis stellen die Frameworks eine Steigerung der Produktivität dar, dadurch eine schnelle Einstieg in den Software-Weltmarkt

5.4 Einführung in Bootstrap und ASP.Net Core

5.4.1 Bootstrap

Bootstrap (Twitter, 2010) ist ein Open-Source Framework, das zur Darstellung von Benutzeroberflächen in einem Web-Browser verwendet wird.

Es erleichtert die Auswahl der Komponenten einer Website und dabei die technischen Anforderungen von verschiedenen Geräten berücksichtigt (Smartphones, Desktops, Tablets) und darüber hinaus gibt das Framework eine solide Grundlage für verschiedene Browseranwendungen.

Die Idee hinter dem Framework ist, die Designer und die Entwickler auf einen gemeinsamen Nenner zusammenzubringen
Bootstrap wurde mit Augenmerk auf HTML5 und CSS3 entwickelt, um so von den aktuellsten Funktionsweisen profitieren zu können, deutlich wird dies beispielsweise bei den in CSS3 eingeführten Eigenschaften für abgerundete Ecken, Farbverläufen und Schatten.

Bootstrap ist relativ einfach aufgebaut. Sein Kern besteht aus verschiedenen Stylesheets, welche die im Framework vorhandenen Komponenten einbeziehen. Über das zentrale Konfigurationsstylesheet können weitere Anpassungen vorgenommen werden.

Vorteile:

- Durch den klaren Leitfaden sparen die Entwickler wie auch die Gestalter Zeit und haben einen gemeinsamen Nenner für die Gestaltung der Oberfläche
- Bootstrap bringt von Haus aus Elemente wie Icons, Boxen, Buttons und PullDown-Menüs bereits mit.
- Erweiterungen wie Modal-Boxen, Tooltips und Tabs sind Teil des Frameworks
- Das Framework ist rückwärtskompatibel. So wird sichergestellt, dass die einzelnen Elemente der Website auf allen Browsern dargestellt werden können.
- Bootstrap ist anpassbar und kann durch Erweiterungen mit neuen Funktionen versehen werden.

Nachteile:

- Das komplette Bootstrap Framework ist ein großer Brocken, was die Ladezeit der Website verzögern kann.

- Durch die vielen Möglichkeiten verleitet das Framework zum Spielen. Effekte oder falscher Einsatz von Scripts führen eher dazu, dass die Website überladen wird.
- Das Layout muss sich dem Framework anpassen.
- Bevor man mit Bootstrap anfängt, sollte man sich mit dem Aufbau des Frameworks befassen und die (meist englische) Dokumentation lesen

5.4.2 ASP.NET Core

Bei Topology-Editor wird die Interaktion zwischen Client und Server durch den Austausch JSON-basierter Nachrichten geschehen, die mittels http-Protokoll übertragen werden. Da Web Services ein Internetdienst ist, müssen die eingesetzten Technologien Plattformunabhängig und unabhängig von einer bestimmten Programmiersprache sein. REST[29] ist eine Möglichkeit, um Web Services zu implementieren. Da wir die C#-Programmiersprache gewählt haben, werden wir als Framework ASP.NET Core 2.0 (Microsoft Corporation, 2018) von Microsoft nehmen.

REST

REST ist plattform-und programmiersprachenunabhängig, und ist einer der verbreisten Möglichkeiten, um Web Services zu realisieren, er ist die Architekturvorbild für das Internet und ist geeignet für die Erstellung von Web Services, und ist kein Standard wie SOAP[30] .
Ein REST-System besteht aus Resources, die per URI[31] adressiert werden (z.B. Ein Device in einem industrielleren Netzwerksystem kann über eine URI adressiert und angesprochen werden)
Rest besitzt folgende Merkmale:

- Die gesamte Nachricht wird in URL kodiert.
- Jede Anfrage muss alle notwendigen Informationen für die Durchführung beinhalten (da http zustandslos ist)
- Darstellungen werden untereinander verlinkt, um dem Client die Möglichkeit zu geben, von einem Zustand in den nächsten zu wechseln.
- Ändert sich die Darstellung einer Ressource oder werden neue Ressourcen zur Verfügung gestellt, kann das Interface des Clients beibehalten werden.
- Konsumiert die HTTP-Methoden wie GET, POST, PUT und DELETE

ASP.NET Core 2.0

ASP.NET Core 2.0 ist ein plattformübergreifendes, leistungsstarkes Open-Source-Framework zum Erstellen moderner, cloudbasierter, mit dem Internet verbundener Anwendungen.
Bei ASP.NET Core 2.0 handelt es sich um eine Neugestaltung Framework von Microsoft mit der Architektur, die ein schlankeres Framework mit größerer Modularität ergeben.

ASP.NET Core bietet die folgenden Vorteile:

- Eine einheitliche Umgebung zum Erstellen der Webbenutzeroberfläche und von Web-APIs
- Integration von modernen clientseitigen Frameworks und Entwicklungsworkflows
- Eine schlanke, leistungsstarke und modulare HTTP-Anforderungspipeline

[29] Representational State Transfer
[30] Simple Object Access Protocol: Protokoll zum Austausch strukturierter Informationen
[31] Uniform Resource Identifier

- Fähigkeit zur Erstellung und Ausführung unter Windows, Open Source und mit Fokus auf der Community
- ASP.NET Core besteht vollständig aus NuGet-Paketen.

6 Topology-Editor Konzept

In diesem Abschnitt wird zuerst die FDT Technologie eingeführt und werden grundlegende Konzepte für den Topology-Editor in Communication Studio vorgestellt, danach werden Kernszenarien dargestellt um zu zeigen wie die Datenquelle erzeugt wird. einschließlich der Kommunikation zwischen Topology-Editor-Client und Topology-Editor-Server. wobei deren Gestaltung in Beziehung zu den in Abschnitt 4.1 und 4.2 behandelten Anforderungen gesetzt wird.

6.1 Topology-Editor grundlegende Konzepte

Topology-Editor ist nicht Teil des FDT2.0-Projekts. In diesem Abschnitt werden die grundlegende Konzepte von dem Plugin erklärt und Leser zeigen, wie die zukünftige Topology-Editor-Plugin implementiert werden könnte. Eine Rahmenanwendung wie Communication Studio kann Topology-Editor unterstützen, um die Datenquelle zu erzeugen.

Topology-Editor-Plugin besteht aus zwei Hauptsoftwarekomponenten nämlich Der Topology-Editor-Server und Topology-Editor-Client.

Der Hauptanwendungsfall für Topology-Editor ist der Offline/Online-Datenaustausch zwischen Geräten und browserfähige Dokumente mit Data-Access-Funktionalität um die Topologie des entsprechenden Systems darzustellen. In diesem Anwendungsfall werden die Gerätedaten von einem Topology-Editor-Server bereitgestellt und von einem Topology-Editor-Client konsumiert, der in das browserfähige Dokument integriert ist.
Topology-Editor bietet Funktionen zum Darstellen einer physikalischen Anordnung (Verknüpfungen falls es zulässt) der protokollabhängigen Geräte mit Datenelementen sowie zum Lesen, Importieren, Exportieren, Löschen, Schreiben und Überwachen dieser Elemente für Datenänderungen.
Die Spezifikationen von Topology-Editor-Server basieren auf der Microsoft ASP.NET Technologie und von Topology-Editor-Client auf dem Google Angular –Technologie für die Kommunikation zwischen Softwarekomponenten verschiedener Hersteller. Daher sind der *Topology-Editor -Server* und die Clients nicht nur auf Windows-basierte Automatisierungssysteme beschränkt aber auch auf Web-Anwendung möglich

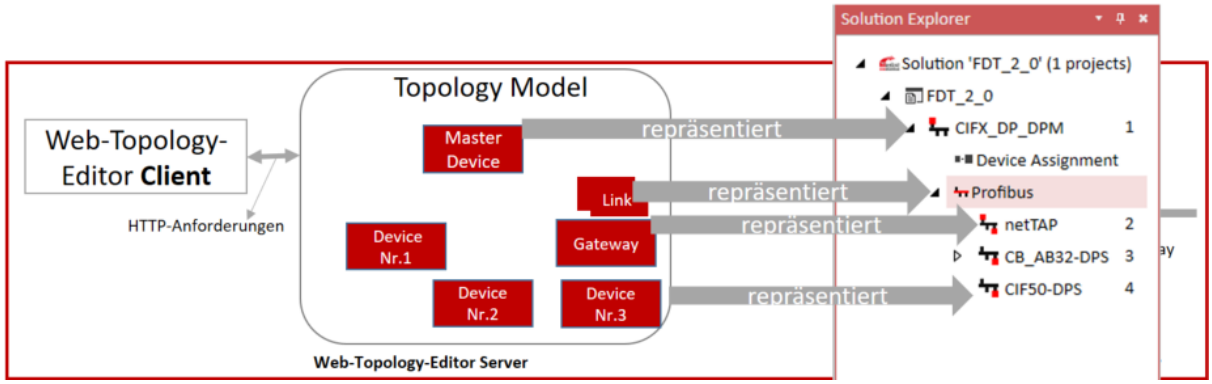
Topology-Editor enthält alle Funktionen der DtmWebApi-Klassenspezifikationen (mehr dazu Abschnitt 5.2) definiert jedoch plattformunabhängige Kommunikationsmechanismen sowie generische, erweiterbare und objektorientierte Modellierungsfunktionen für die Informationen, die ein System bereitstellen möchte.
Der Topology-Editor -Netzwerkkommunikationsteil definiert Mechanismen, die für verschiedene Anwendungsfälle optimiert sind.
Die entstehende Version von Topology-Editor wird ein optimiertes HTTP-Protokoll für hochperformante Intranet-Kommunikation sowie eine Abbildung auf akzeptierte Internetstandards wie Web Services definiert. Das abstrakte Kommunikationsmodell hängt nicht von einer spezifischen Protokollzuordnung ab und ermöglicht das Hinzufügen neuer Protokolle in der Zukunft. Funktionen wie Sicherheit, Zugriffskontrolle und Zuverlässigkeit sind direkt in die Transportmechanismen integriert.

Das Topology-Editor -Modell bietet eine Standardmethode für Server, um Objekte Clients zur Verfügung zu stellen. Objekte können sich aus anderen Objekten, Variablen und Methoden zusammensetzen. Topology-Editor erlaubt auch, Beziehungen zu anderen Objekten auszudrücken.
Die Menge der Objekte und zugehörige Informationen, die ein Topology-Editor-Server Clients zur Verfügung stellt, werden hier als Topology bezeichnet. Die Elemente der Topology-Editor -Objektmodells werden im Topology als eine Gruppe von Knoten dargestellt, die durch Eigenschaften beschrieben und durch Referenzen miteinander verbunden sind.

Topology-Editor definiert drei Klassen von Knoten zur Darstellung von Topology -Komponenten. Die Klassen sind Objekte, die Geräte, Controller und Kommunikationsinfrastrukturgeräte repräsentieren. Jede Knoten-Klassen und Verbindung-Klassen haben einen definierten Satz von Eigenschaften und Methoden.

In einem Multiclient-Szenario ist die Communication Studio für die Verwaltung des Zugriffs auf die DTMs zuständig.

In diesem Abschnitt werden die häufigsten Szenarien beschrieben, die bei der Arbeit mit Topology-Editor in Communication Studio auftreten. Diese Szenarien umfassen das Offline-Konfiguration eines Projekts, das Registrieren und Erkennen von Geräte, das Laden von XML-Topologie, das Erkennen, wenn ein Device eingefügt bzw. gelöscht wurde. Sie können im Offline-Module ein gerätspezifischer Device in einer FDT2.0-Projekt oder durch Scannen eines lokalen Verzeichnisses registrieren und erkennen.



In Laufe der Applikation wird durch einen sogenannten Scan-Mechanismus der spezifischen Anordnung der Geräte, die untereinander verbunden sind, generiert.

Die generierten Daten werden als JSON-Datei dargestellt. Der Zweck einer JSON-Datei besteht darin ein Diagramm zu definieren, das ein Netzwerk darstellt.

6.1.1 ComStudio Scenario

6.1.2 Schnittstellen zwischen Komponenten

6.1.3 Daten-Handling-Konzept

6.2 User interface

6.2.1 ComStudio Scenario user interface

6.2.2 Main operation user usercontrol für Toplogie

6.2.3 Ribbon mit Register Topologie

6.2.4 Solution Explorer View

6.3 Datendarstellung (Prozessdaten, Bedienungsdaten)

Das Konzept der Topology-Editor soll prinzipiell die physikalische Netzstruktur einer industriellen Kommunikation auf einer Schnittstelle (Laptop Mobile Geräte) bereitstellen.

Um grundlegende Konzepte erklären zu können, betrachten wir in dem folgenden Abschnitt eine Mini-Topologie, die nur aus 2 Knoten und eine Kante besteht.

Es enthält 2 Knoten und eine Kante: jeden Knoten repräsentiert ein Device und eine Kante eine spezifische Verbindung. Die Mini-Topologie wird durch die folgende JSON-Datei repräsentiert.

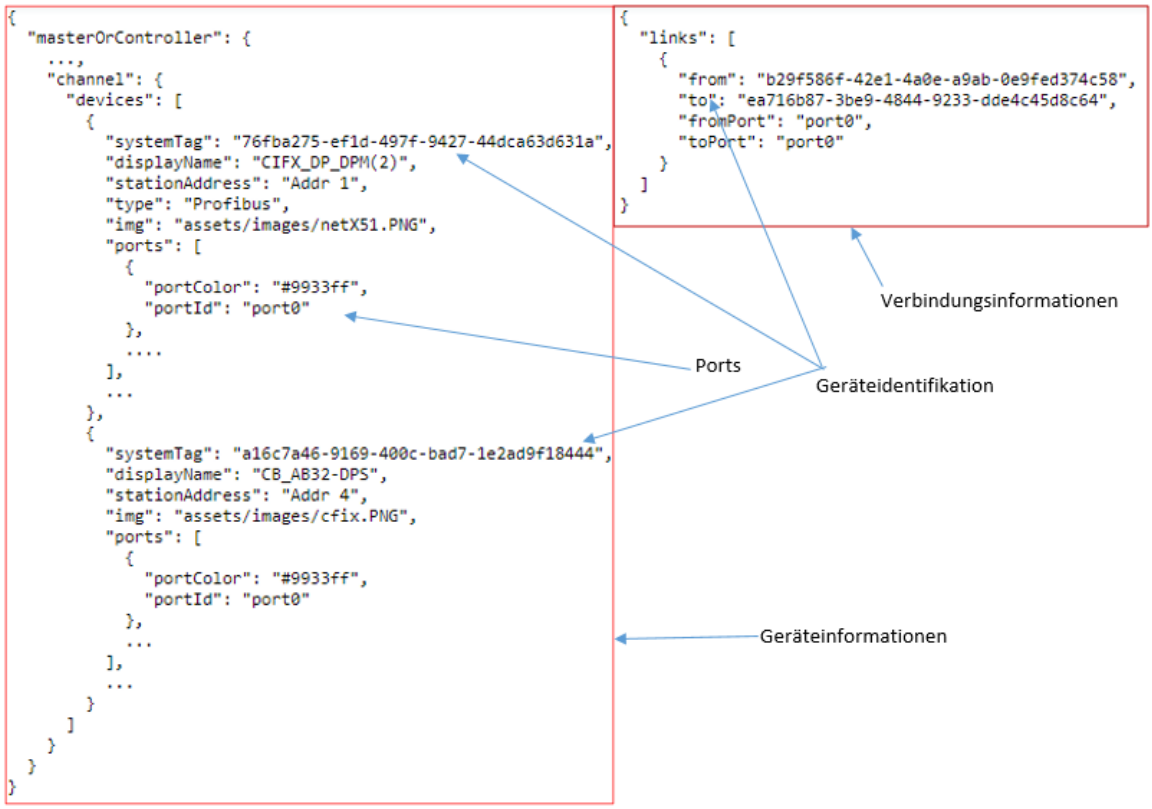


Abbildung 15: Die Output JSON-Dateien für die Mini-Topologie

Das Output des Scan Mechanismus besteht aus einen Root Element und einer Vielzahl von Unterelemente: Graph, Knoten, Kante. Im weiteren Verlauf dieses Abschnitts werden wir diese Elemente im Detail besprechen und zeigen, wie sie ein Diagramm definieren.

Die Topologie-Struktur, die Knoten und Kanten enthält, wird als Graph bezeichnet. Ein Graph besteht aus zwei JSON Dateien nämlich:

- Geräteinformationen: In einem Geräteinformation JSON-Datei sind die Deklarationen von Knoten mit dem Schlüsselwort „*devices*“ und Eigenschaften der Topologie als Root-Elemente definiert. Die Liste von Ports von jeweils Geräte wird innerhalb eines – device-Element mit dem Schlüsselwort „Ports“ definiert. und
- die Verbindungsinformation: hier wird die List von Verbindungen mit dem Schlüsselwort „links“ definiert und jede link-Element repräsentiert eine Verbindung zwischen 2 Geräten. Durch das Schlüsselwort „*from*“ wird das Quellgerät identifiziert und „to“ das Zielgerät identifiziert bzw. „formPort“ wird das Port des Quellgerät und „toPort“ Zielgerät festgelegt.

Knoten repräsentieren die Geräte im Graphen und werden mit Hilfe vom Schlüsselwort „devices“ aufgelistet Jeder Knoten hat eine Kennung, die innerhalb des gesamten Dokuments eindeutig sein muss, d. H. In einem Dokument dürfen keine zwei Knoten mit der gleichen Kennung vorhanden sein. Die Kennung eines Knotens wird durch „*systemTag*“-Schlüsselwort definiert, die eine Zeichenfolge ist. Jeder Knoten muss ein „*displayName*“-Wert haben, der den Namen von der Device bereitstellt (Siehe Abbildung 10) und „stationAddress“-Wert haben der die Gerätadresse definiert.
Die andere Option können optional sein wie (Ports, Bilder, Range ...)

Die Verbindungen werden durch Kanten im Graphen repräsentiert. Jede Kante muss mit Hilfe der „link“-Schlusswort aufgelistet werden. Sie muss ihre zwei Endpunkte mit der Quelle und dem Ziel definiert sein. Der Wert der Quelle bzw. Ziel muss der „*SystemTag“* eines Knoten sein. Genauer so soll der Wert der Quelle-Port bzw. Ziel-Port der „*portId*“ eines Knoten sein

Jede Kante kann ein optionales Schussel-Label haben, das eine Zeichenfolge ist.

Das Gewicht der Kante wird durch das optionale Schlüsselwort Gewicht festgelegt und ist ein Float.

6.4 Zukunftsaspekte (evtl. Erweiterungen)

Da die Physikalische Topologie ein wachsendes Thema der Forschung in der industriellen Kommunikation ist, werden wir in der Zukunft ein eigenes Format definieren, für das Output des Scan-Mechanismus die sogenannte TOPO Dokument.
Das TOPO-Dokument wird ein XML-Dokument basiert sein.

In der Spezifikation der TOPO-Dokument soll folgendes definiert sein:

- Wie ein Graph mit Hilfe von XML-Elemente und XML- Attributes definieret werden soll
- Wie Knoten und Verbindungen mit Hilfe von XML-Elemente deklariert werden sollen.
- Das Dokument soll als TOPO Datei gespeichert werden.
- TOPO soll die XML-Schemadatentypen XSD 1.1 für die Grundelemente wie ganze Zahl, Datum, Zeichenkette verwenden

Es wird auch eines Mechanismus für Übertragungssicherung in die Zukunft geplant

Wie schon in der Einleitung erwärmt wird, wird auch Echtzeitverhalten während der Kommunikation zwischen Komponente geplant.

Für die Zulässigkeit einer Topologie Arte wird auch eine eigene Module geplant. Eine der wichtigsten Zukunftsaspekte ist die logische Topologie für alle feldbusabhängige Industrielle Netzwerk.

7 Entwurf

In diesem Kapitel wird das zu realisierende Plugin entworfen. Hierfür wird zunächst auf die Architektur Topology-Editor eingegangen und anschließend das Datenmodell erläutern. Ausgehend von den erhobenen funktionalen und nicht funktionalen Anforderungen in Abschnitt 7.1 und 7.2 entsteht der Entwurf.

7.1 Architektur der Topology-Editor

Die Architektur der Topology-Editor basiert auf die Architektur einer SPA[32] denn der SPA-Ansatz reduziert die Zeit die von der Anwendung für die Reaktion auf Benutzeraktion , was eine mehr flüssige Erfahrung ist.

In herkömmlichen Webanwendung verarbeitet der Server die Anforderung und sendet Sie im Browser als Reaktion auf die neue Aktionsanforderung von Client, jedoch bei SPA muss der Browser nur der Bereich auf die Seite zu aktualisieren, die geändert wurden; Es ist nicht erforderlich, die gesamte Seite erneut zu laden.
Die Architektur von Topology-Editor umfasst einige Herausforderungen, die nicht in herkömmlichen Webanwendung vorhanden sind. Allerdings begegnet Technologien wie ASP:NET:Web-API, JavaScript-Framework wie Angular 5(Abschnitt 6.1) und CSS-Framework wie Bootstrap(Abschnitt 6.2) können entworfen werden und Aufbauen von SPAs wirklich vereinfachen.

[32] Single Page Application

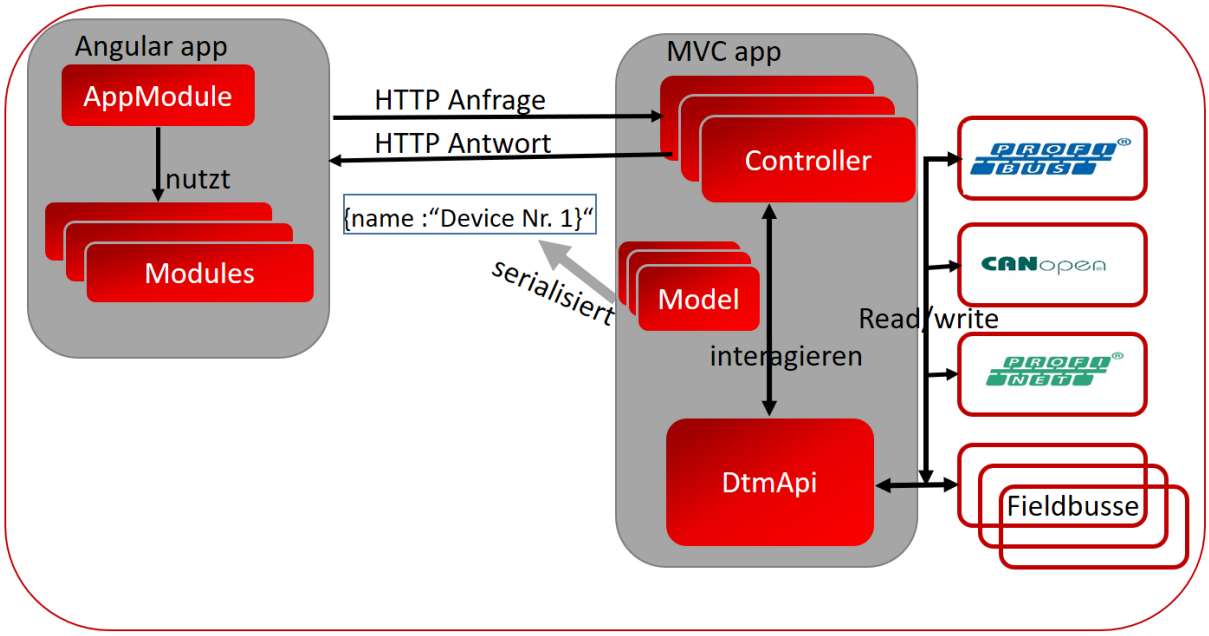


Abbildung 16: Diagramm zeigt den Grundlegenden Entwurf der Topology-Editor

Die Abbildung zeigt die den Grundentwurf der Topology-Editor und wird im Folgenden erklärt. Der Client ist das Medium, das die Web-API nutzt (mobile App, Browser usw.). Hier wird der Client als Angular App erstellt. Das Angular App besteht aus 4 Hauptmodulen und werden als TypeScript-Klassen dargestellt.
Die Modelle sind Objekte, die die Daten in der App darstellen. In diesem Fall sind drei Hauptklassen erstellt Modelle werden als C#-Klassen dargestellt, die auch als Plain Old C# Object (POCOs) bezeichnet werden.
Controller sind Objekte, die HTTP-Anforderungen verarbeitet und die HTTP-Antwort erstellt. Sobald eine Anfrage des Clients mit der URI eingeht, wird die verlinkte Funktion in dem betreffenden Controller aufgerufen. Die Controller repräsentieren die Anwendungslogik der Applikation. Sie können mit den Modellen und Wrapper (DtmApi) interagieren, um insbesondere Daten abzufragen.
Den Wrapper DtmApi ist die Schnittstelle für die feldbusprotokollabhängigen Parameter und die Einstellungen die hier an der Stelle nicht betrachtet werden sonders als gegeben vorausgesetzt werden.

7.2 Datenmodell

Aus Gründen der Übersicht wird in diesem Abschnitt nur auf die für die Plugin besonderes relevanten Entitäten der Topology-Editor eingegangen (Abbildung 17). Eine vollständige Übersicht befindet sich im angehängte CD.
Aus den funktionalen Anforderungen ergibt sich direkt das Domänenmodell.

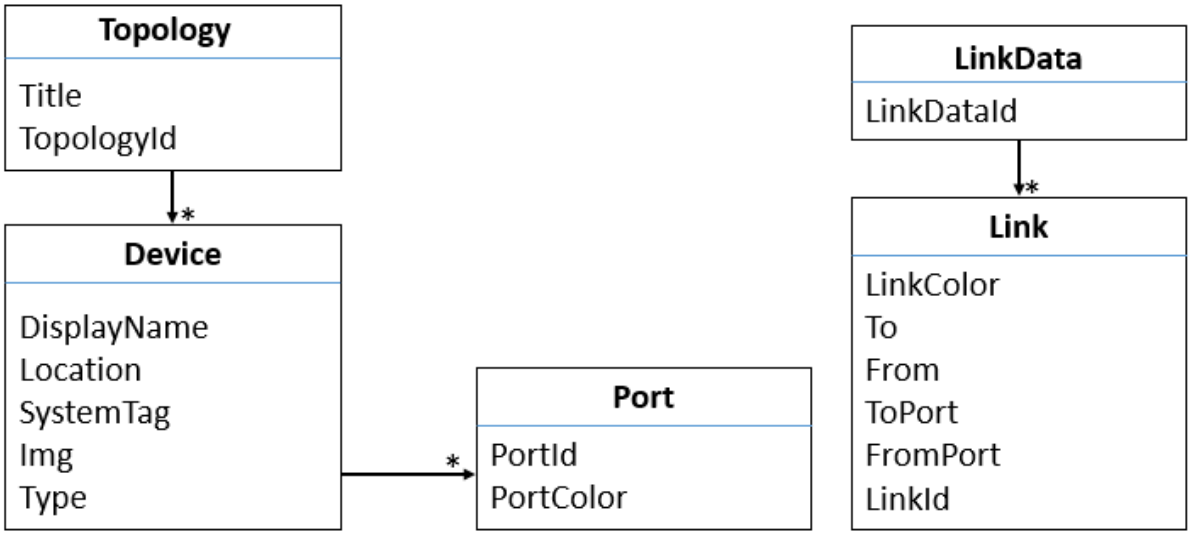


Abbildung 17: Domänenmodell der Elemente des Topology-Editor

Das Entität *Topology* repräsentiert die Topologie des bereitgestellten industriellen Netzwerks ohne Berücksichtigung der Verbindungen und enthält wesentliche Merkmale wie etwa die Überschrift *Title*, Die Identifikationsnummer *TopologyId* und vor allen die Liste der Geräte die sich in Netzwerk befinden.
Jede Device kann wiederum mehrere Ports besitzen; Ein Port entspricht den Zugangspunkt einer Device und wird von dem Entität *Link* verwenden um die Verbindung herzustellen
Das Entität *LinkData* ist verantwortlich für die Verbindungen einer Topologie zu listen und die dazugehörige Topologie zu identifizieren

7.3 DtmApi Wrapper

DtmApi besteht aus zwei Hauptfassaden nämlich TopologyFassade und LinkFassade, die Methoden und Eigenschaften bereitstellt, die eine Untermenge der Funktionalität des Systems darstellen. Andere Klassen greifen nur noch auf diese Fassaden zu. Dadurch wird die Benutzung der Gruppen von Klassen (sind meisten feldbusprotokollabhängig) einfacher und Veränderungen hinter den Fassaden bleibt ohne Auswirkungen auf die Nutzer der Fassaden. Als Nebeneffekt wird die Funktionalität für die Entwickler leicht zu verstehen, da sie nur noch die Fassaden und nicht die Details der Gruppe der feldbusprotokollabhängigen Klassen verstehen müssen. Man kann sagen, dass diese Fassaden die Abstraktion der Gruppe darstellen.

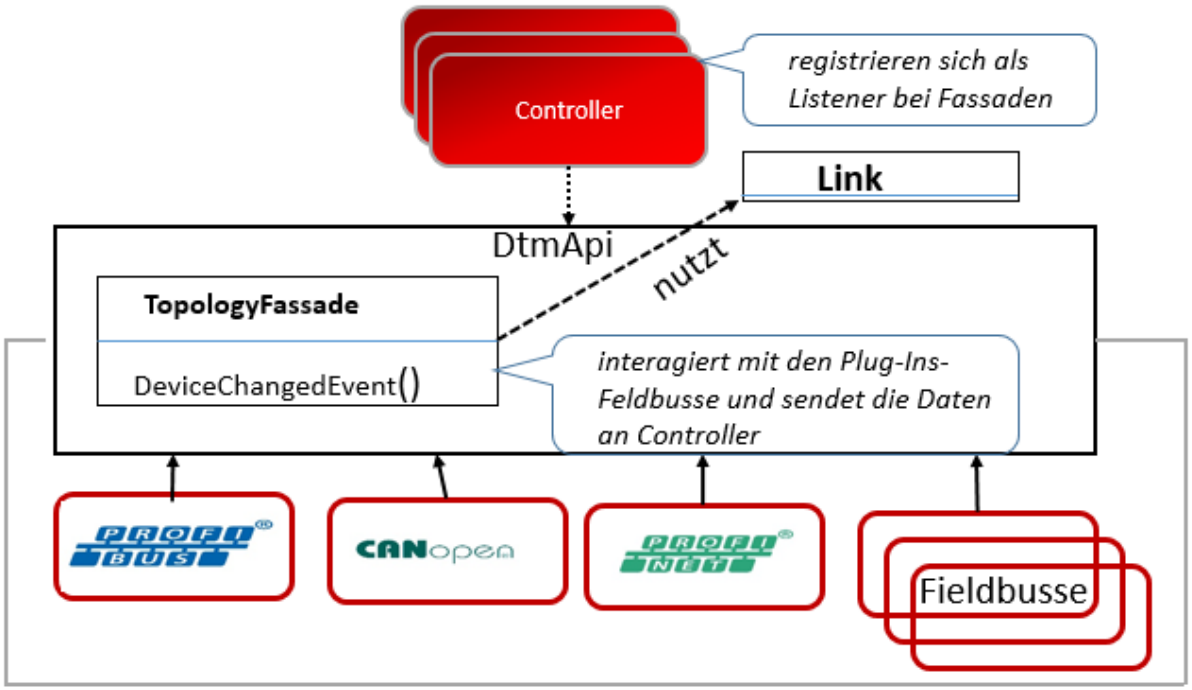


Abbildung 18: Klassendiagramm eines Beispiel für Fassaden mit feldbusabhängigen Protokollen

Abbildung 18 zeigt Controller, die Fassaden als Beobachter für die Tätigkeiten wie (zum Beispiel die Add, Remove und Change Device) angemeldet sind. An welchen Konkreten Feldbusklassen sie sich tatsächlich anmelden oder wie dieses Feldbusse intern aufgebaut sind, spielen so für Controller keine Rolle.
Auf Grund der Übersicht wurden die Fassadenklassen sichtbar mit nur jeweils eine Methode gemacht (schwarz fett), um die wichtige Rolle der Wrapper zu erklären.
Die Plugins-Feldbusse sind Runde Rechtecke mit rote Rahmen.

8 Realisierung

In diesem Kapitel wird kurz auf die Punkte der Implementierung des Plugin Topology-Editor eingegangen. Hierzu gehört die Auswahl der verschiedenen Programmiersprachen und die verwendeten Werkzeuge.

8.1 Vorgehensweise

Beim Unternehmen Hilscher ist das *Scrum*[33]-Vorgehensmodell in der Abteilung Group User Interface für die Durchführung von Softwareprojekten vorgeschrieben. Da es sich um ein Plugin handelt, wird ein iterativ inkrementelles Vorgehensmodell eingesetzt. Dadurch können die implementieren Module schrittweise in die bestehende Software integriert werden.

8.2 Verwendete Werkzeuge

In diesem Kapitel werden die verwendeten Technologien und Werkzeuge angesprochen. Auch nichtfunktionale Aspekte wie die Wiederverwendbarkeit des Quellcodes oder der bereits verwendeten Technologien in der Firma spielen eine Rolle. Für die Realisierung des Plugins wurde als Haupterkzeug für die serverseitige Umsetzung die Entwicklungsumgebung Microsoft Visual Studio 2017 (Microsoft , 2018) mit der Programmiersprache C# gewählt, für clientseitige Umsetzung der Visual Studio Code (Microsoft, 2018). Des Weiteren werden Bootstrap und ASP:NET Core verwendet, wie bereits in Abschnitt 5.4 erläutert

8.3 Ausgewählte Implementierungsaspekte

In diesem Kapitel werden ausgewählte Teile der Implementierung vorgestellt. Hierbei stehen die Erfassung der Nutzereingaben und die daraus resultierenden Verarbeitungsschritte der entsprechenden Daten bis zur Anzeige der Topologie im Vordergrund. Zunächst wird in Kapitel 8.3.3 darauf eingegangen, wie verschiedene Geräte- und Verbindungsinformationen übertragen werden.

Anschließend wird in Kapitel 8.3.4 gezeigt, wie anhand dieser Informationen eine Topologie aktualisiert wird und geeignete Daten gefunden werden. Danach wird betrachtet, wie mit Hilfe der gefundenen Daten die Topologie erstellt wird. Abschließend wird die grafische Umsetzung der Darstellung und die Auswertung erläutert.

[33] ist die Bezeichnung für ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung.

8.3.1 Realisierung der Server-Applikation

Der Server ist ASP.Net Core basiert, mit der Vorlage Web-API. Er besteht aus drei wesentlichen Komponenten nämlich Hilscher.TopologyEditor, Hilscher.TopologyEditor.AspNet und die Hilscher.Web.API

Diese drei Komponente unterstützen die Software ComStudio dabei, alle Netzwerk-Protokolle für Web-Services mithilfe des Model-View-Controller-Musters zu kommunizieren. Sie wurden jedoch momentan hauptsächlich für PROFINET IRT entwickelt, die aus einzelnen, funktional vollständigen Teilen bestehen, die zusammen eine einzelne integrierte Schnittstelle bilden. Die TopologyEditor-Library beschleunigt die Entwicklung von Anwendungen mit bewährten Entwurfsmustern.

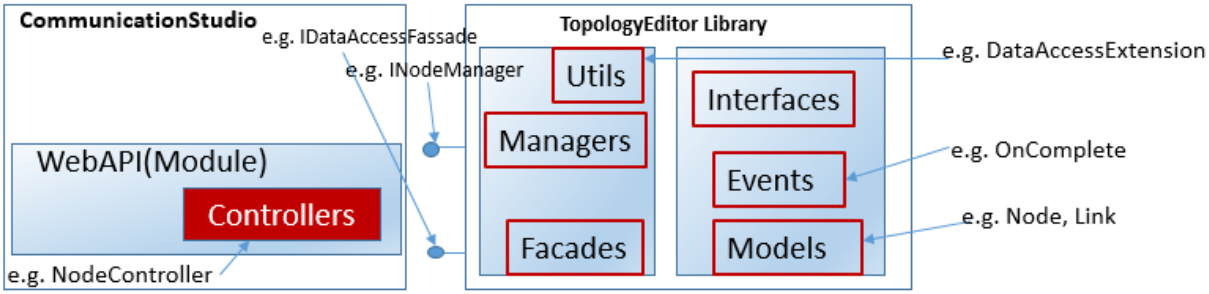


Abbildung 19: Zusammengesetzte Anwendungsarchitektur mit häufig verwendeten Paketen

Hilscher.TopologyEditor enthält die Kernfunktionalität von TopologyEditor und ist für die Integration der feldbusprotokollabhängigen Parameter. Das beinhaltet:

- **Modellklassen**: wie Node, Link, Toplogy, die unter anderem die spezifischen DTMs und Einstellung repräsentieren.
- **Events** stellt einen Ereignishandler bereit, der das Auftreten einer Aktion wie „Das Laden von Parameter beenden“ signalisiert.
- **Interfaces**: stellt die Schnittstellen zur Kommunikation mit physikalischen Geräten beziehungsweise DTM-Geräten bereit. Zu dieser Schnittstelle gehören beispielerweise IDataAccess, IDtmDataAccess. Außerdem definiert sie die Schnittstellen INodeManger und ILinkManger zur Verwaltung von Node und Link

Hilscher.TopologyEditor.AspNet stellt Klassenkomponente bereit, um ASP.NET mit TopologyEditor Library zu verwenden. Zu diesen Komponenten gehört die tatsächliche Implementierung von allen Schnittstellen von Interfaces und vieler Hilfeklassen

- **Facades** sind für die Konkrete Klassen zur Kommunikation mit DTM-Geräte verantwortlich, darin befindet sich Beispiel TopologyDataAccess
- **Managers**: beinhaltet die konkreten Implementierungen von Managers-Schnittstellen aus Interfaces. Dazu gehören NodeManger, LinkManager
- **Utils** umfasst statischen Klassen, die das Erstellen von Modellkassen erleichtern sollen und definieren statische schreibgeschützte Eigenschaften, um spezifische Parameter zugänglich zu machen.

Hilscher.TopologyEditor.Web.API ist eine Sammlung von Controllern, die HTTP-Anforderungen verarbeitet und entsprechende HTTP-Antworten generiert. Sie verwendet die zwei vorgestellten Assemblies. Diese Komponente hat zwei Controller, der eine verarbeitet die Device-Anfragen und der andere die Verbindungen-Anfragen.

8.3.2 Realisierung der Client-Applikation

Wie bereits erwähnt, haben die benutzten Technologien, die zum Teil nach der Evaluation der JavaScript-Frameworks gewählt wurden und zum Teil durch die Problemstellung vorgegeben waren, beeinflussen sehr stark das Endprodukt und Entwicklung hinsichtlich der Client-Applikation.
Um einen Einblick in das Software-Modul Client-Topology-Editor selbst zu geben und aufzuzeigen wird das Modul-Diagramm mit Klassen von Client-TopologyEditor besprochen und dabei eine Übersicht über die Rolle von JSON-Dateien als Informationsquelle von Topology-Editor gegeben.

Zur Veranschaulichung der eigens für Client-Topology-Editor und der aus Angular-Framework und GoJs-Bibliothek verwendeten Klassen wurde das Diagramm in Abbildung 20 erstellt. Es zeigt das AppModule in der Mitte, um das sich alle Klassen von Client Topology Editor anordnen. Als Einstieg in das Diagramm können die Komponente-Klassen und Module-Klassen angesehen werden. Angular-Komponenten repräsentieren jeweils eine Ansicht in der App. Sicherlich startet man die Client-App als User nicht in irgendeiner Ansicht, sondern mit der Ansicht zum Editor, der „Topology Editor.html“.
Wie bereits angedeutet, wird eine Ansicht einer Angular-App zunächst als Component implementiert, dem man ein Template und ein CSS-File zuweist. Sobald das System eine Ansicht anzeigen soll, wird der Hauptteil der Funktionalität zu entsprechenden Service delegieren, da in Service die Hauptteile der Funktionalitäten umgesetzt wurden. Dadurch werden die Anwendungslogik und Präsentationlogik getrennt.

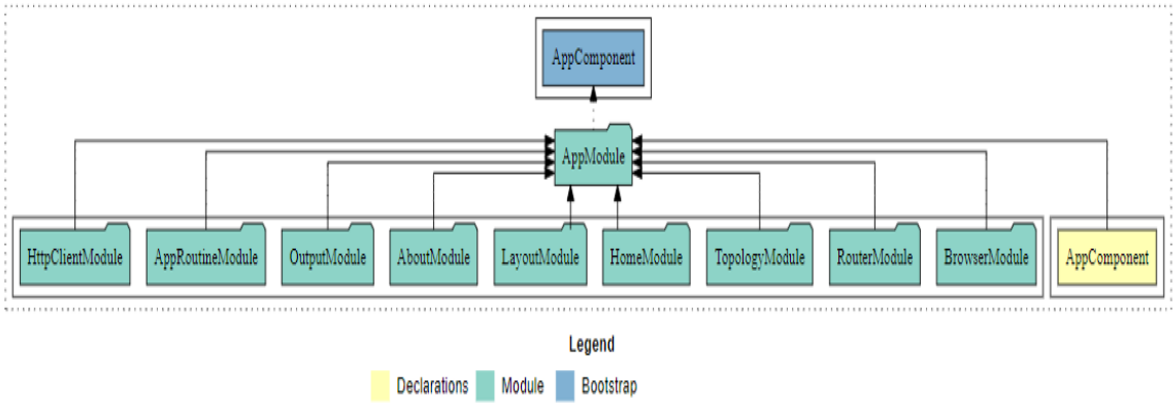


Abbildung 20: Vereinfachtes Diagramm für die Umsetzung der clientseitigen Anwendung

Die Ganze Funktionalität, die einen Einfluss auf den Ausbau und die dynamische Anpassung der Ansicht hat, wird in Topology Editor unter einem Thread ausgeführt. Das ist deshalb nötig, weil die Kommunikation mit UI-Elementen nur auf dem Hauptthread auch UI-Thread genannt.

Manche Anweisungen, wie zum Beispiel http-Anfragen, können mitunter zu lange dauern und den UI-Thread blockieren. Um diese zu umgehen, müssen diese Funktionen in einem vordefinierter Verfahren von Angular-Framework ausgelagert werden. Dieses Verfahren wurde zum Senden und Empfangen der Topologie-Informationen im System im Rahmen des CRUD[34]-Operators in NET.Core 2.0 (Microsoft, 2018) umgesetzt.

Weitere Funktionalität, die auch über das Beenden der Plugin hinaus agieren kann, wird unter Angular als Angular Resolver (Google, 2010) implementiert. Die Angular Resolver wird hauptsächlich von der Komponentenklasse verwendet. Dennoch ist ein Resolver nicht zum Auslagern aufwändiger Berechnungen geeignet, sondern muss als eigenständiger Teil des Programms angesehen werden.

Sofern ein neues DTM in das FDT-Projekt hinzugeführt wird, schickt der ComStudio eine Notifikation an der Client-App, die darauf hinweist, dass es ein neues Device (repräsentiert durch ein DTM) in System gibt; mit Hilfe des Service kann der Client der Topology Edithor.html aktualisieren. Diese Anweisung wird in der updateTopology()-Methode umgesetzt.
Die am häufigsten verwendeten Services sind.

- „TopologieService.ts“ : liefert die Topologie der gewünschten Netzwerke als ein Objekt, welches GoJs-Bibliothek verwenden kann.
- „DeviceService.ts“: liefert die Liste der Geräte und wird von TopologyService konsumiert
- „LinkService.ts“: liefert die Liste der Verbindungen, die zur gewählten Topologie gehört.

Für die Kommunikation mit den beiden JSON-Dateien sind noch die Klassen „DeviceService.ts“ und „LinkService.ts“ verantwortlich. Sie verfügen über einem großen Satz an Methoden, um entsprechende JSON-Dateien im System zu manipulieren, weshalb sie eine zentrale Rolle in den Modulen spielen. Die beiden Klassen sind im Singleton-Pattern[35] erstellt worden und wurden in entsprechenden Komponenten integriert.

8.3.3 Systemvorbereitung und Erstellen eines FDT-Projekt

Um das Plugin zu nutzen, muss der Nutzer vor allem die Software ComStudio starten. Dabei wird der Server für das TopologyEditor automatisch gestartet. Der Nutzer legt ein FDT-Projekt (siehe Abschnitt 6.1) über einen vordefinierten Mechanismus der ComStudio an. Danach wird einige Geräte in diesem Projekt eingeführt. Nach dem Beenden dieser Schritte werden die getätigten Eingaben gespeichert. Dafür wird das C# Objekt *IProject* verwendet, das in ComStudio vordefiniert wurde. Dieses C# Objekt bietet den Vorteil, dass es einfach über die Methode Build() zu einer ITopologyRoot konvertiert und so übertragen werden kann.

[34] CRUD steht für Create, Read, Update, Delete
[35] Ein Singleton-Pattern sorgt dafür, dass es von einer Klasse nur eine einzige Instanz gibt und diese global zugänglich ist

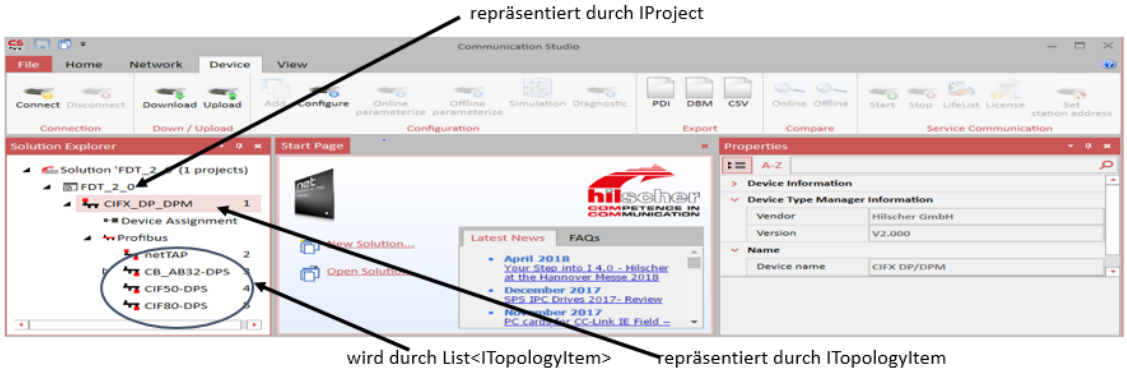


Abbildung 21: Erstellen eines FDT-Projekt

Um die Eingaben des Nutzers für das Erstellen des FDT-Projekt speichern zu können und später die Topologie darzustellen, bietet das IProject verschiedene Eigenschaften an, wie systemTag, Items und weitere. Diese Eigenschaften repräsentieren die Filter, nach denen TopologyEditor Server ein Topologie-Modell erstellen kann. Eine besondere Rolle spielen hierbei die Eigenschaften Children vom Typ List<ITopologyItem>, Da sie die Liste der Root-Geräte darstellt. Hierfür wird jedes Element der Liste an das Topology-API übergeben, um das Topologie Model zu konstruieren. Die Eigenschaft *systemTag* das Element identifiziert eindeutig jedes Element in dem gesamten System. Er ist als GUID-String codiert.

8.3.4 Anzeigen einer Topologie

Nachdem wir das System initialisiert haben, können wir nun die Software nutzen. Wir werden nur die Topologie für einzelne Master-Geräte darstellen nicht abe für das gesamte Projekt da es nicht Teil der Aufgabe ist.
Der Nutzer kann entweder das TopologyEditor-Client in einem Browser laufen lassen oder direkt in ComStudio, siehe Abbildung, öffnen.
Die Rohdaten werden nun vom Server als http-Antwort an den Client gesendet. Es werden zwei Antworten parallel geschickt: Die Rohinformationen für die Liste der beteiligten Geräte und für die Liste der dazugehörigen Verbindungen. Der Client empfängt diese Informationen als JSON-Objekte, bereitet sie mit Hilfe von HTML und TypeScript grafisch auf und stellt sie dar. Zur Darstellung der Symbole wird die CSS-Bibliothek GoJs (Siehe Abschnitt 5.3) verwendet.

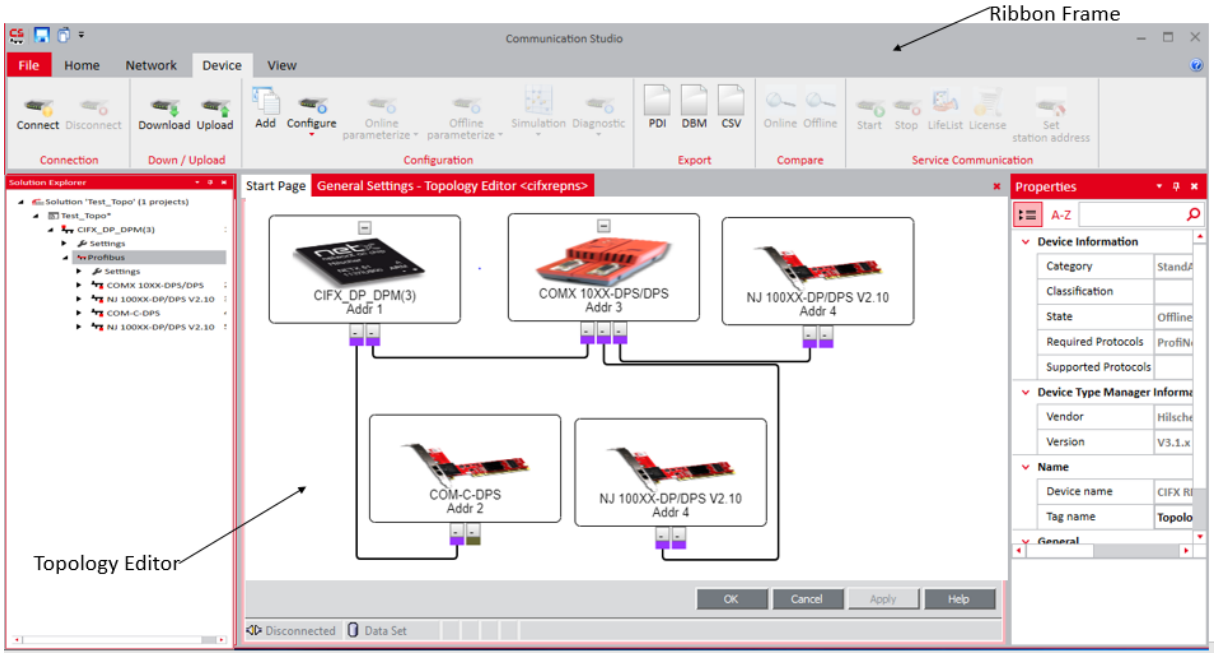


Abbildung 22: ComStudio mit Topology Editor im Einsatz

Der Benutzer muss im Editor zum Erstellen und Anpassen einer Topologie je nach Anzahl der Devices scrollen, um alle Information einsehen und bearbeiten zu können. In der Abbildung ist die detaillierte Information über ein Device bzw. eine Verbindung rechts im Fenster Properties dargestellt.

9 Zusammenfassung und Ausblick

Abschließend werden in diesem Kapitel die vorgestellten Inhalte zusammengefasst und Ansatzpunkte für die zukünftige Weiterentwicklung des Topologie-Editor in einem Ausblick gegeben.

9.1 Zusammenfassung

Die ursprünglich gesetzten Ziele mit PROFINET IRT sind teilweise erreicht. Nach einer Übersicht über alle Topologie-Arten, wurde der Ist-Zustand hinsichtlich der ComStudio dargestellt und die Anforderungen der Plugin analysiert. Dabei konnten mehrere Schwachstellen bezüglich der Konfiguration-Prozess mit dem bestehenden Konfigurator feststellen.
Die Analyse der logischen Verbindungen und der Echtzeitverhalten waren nicht Bestandteile dieser Arbeit, daher wurde nicht untersucht.

Der Aufbau des Editors für die Topologie erlaubt es, dem Inbetriebnehmer eines Netzwerks einfach und mit wenig Aufwand seine Aufgaben, das Netzwerk zu konfigurieren. Durch die Repräsentation der Geräte als Rechtecke mit gezielte Informationen in dem Graphen, ist es ein Einfaches für den Hilscher Mitarbeiter auch Kunden daran mitarbeiten zu lassen.
Die bereits umgesetzten Linien(Verbindungen) bieten eine Basis wie die feldbusabhängigen Geräten in dem Netzwerk gebunden sind. Dadurch können den Kunden auf schnellem Weg die Beziehung zwischen Teilnehmern feststellen. Aus dieser Sicht ist das Software Plugin-Modul Topologie-Editor vollkommen geeignet für eine Kostenreduktion und darüber hinaus liefert ein „Look and Feel“-Effekt.

Da der Mensch diese Art der visuellen Informationen am schnellsten Erfassen und Verstehen kann, ist es sicherlich von Vorteil für den Einsatz beim Projektieren eines Feldbussystems.

Die Erstellung von Desktop-Anwendungen und Web-Anwendungen aus einer gemeinsame Codebasis lässt sich mit dem JavaScript-Framework Angular 2 umsetzen. Die Realisierung mit diesem Framework erlaubt im Verglich mit den anderen Optionen aus Kapitel 5 den größten Grad der Wiederverwendung des Quellcodes für die Plattformen und Webbrowsers.
Zur Implementierung wird ein einheitliches Set an Technologien verwendet, plattformspezifische Technologie wie Beispielerweise JSON werden öfter benötigt als XML-Markup die Codebasis erreicht daher den größtmöglichen Grad an Einheitlichkeit.

Durch die Wiederverwendung des Quellcodes für Desktop-Anwendung und Web-Anwendung fällt insgesamt ein geringerer Entwicklungs- und Wartungsaufwand an. Das Framework Angular bietet zudem eine Reihe von Konzepte zur Strukturierung des Quellcodes welche sich über

die verbesserte Modularität und Wiederverwendbarkeit positiv auf die Wartbarkeit der Codebasis auswirken. Neben diesen Konzepten aus dem Abschnitt 5.1.1 gibt es auch das MVC-Architekturmuster eine bestimmte Struktur während der Kodierung vor. Dadurch müssen von Entwicklerseite keine eigenen Konzepte zur Strukturierung entwickelt und eingehalten werden, sondern es wird eine bestimmte Anwendungsarchitektur vorgegeben.

Durch die Verwendung von Visual Studio Code (Siehe Abschnitt 8.2) mittels Transpiler lassen sich viele Probleme der Programmiersprache JavaScript beheben und die Syntax unter anderem um Sprachkonstrukte für klassenbasierte Objektorientierung erweitern. Dies verringert den Wartungsaufwand von JavaScript-Anwendungen und ermöglicht Entwicklern einen leichteren Zugang zur objektorientierten Entwicklung mit JavaScript und damit eine bessere Modularität der Systems.

TypeScript verbessert die Wartbarkeit von JavaScript zusätzlich, indem dieses um eine statische Typisierung ergänzt wird. Ein zusätzlicher Kompilierschritt ermöglicht die Prüfung des Quellcodes, um den Entwickler im Umgang mit fehlerträchtigen Eigenschaften der Programmiersprache zu unterstützen, beispielsweise der permissiven Fehlerbehandlung oder automatischen Type Konvertierungen. Dies verbessert in erster Linie die Analysierbarkeit und dadurch die Änderbarkeit der Codes.

9.2 Ausblick

Während der Entwicklung des Plugins wurde deutlich, dass vor allem die Zeit zwischen Anfragen und Antworten ein nicht zu unterschätzendes Problem ist. Dauert die Verarbeitung der feldbusprotokollabhängigen Informationen zu lange und die Antwort wird als Resultat dessen nicht gesendet, läuft der Browser beziehungsweise browserfähige Control unter Umständen in einer sogenannte *Connection Timeout*. Diese geschieht im Besonderen dann, wenn insgesamt betrachtet viele Geräte unter viele Kriterien zu einem Master-Gerät herangezogen werden. Das Problem wird mit der Zeit immer größer, da die Anforderungen an Netzwerke die aus Industrie kommen, steigen stetig. Obwohl wie ich schon in der Einleitung erwähnte habe, wird die Übertragungsgeschwindigkeit der Daten nicht betrachtet, muss eine Lösung je schneller desto besser gefunden werden.
Das Einführen des Single-Page App (Siehe Abschnitt 7.1) war ein erster großer Schritt in die richtige Richtung. Sie spart viele tausende Abfragen der Daten pro erstellter Netzwerk-Topologie, was bisher den zeitlichen Aufwand ungefähr halbiert hat.

Bei einer Topologie geht es vor allem darum, die Verknüpfung der Geräte ein industrielles Netzwerk darzustellen. Daher wäre n weitere Darstellungsmöglichkeiten, wie etwa eine logische Darstellung des Datenflusses zu legen sinnvoll. Ebenfalls könnten die Anzeige der Geräte in einer Kommunikationsinfrastrukturgerät (z.B. Gateway) ergänzt werden.

Man könnte auch die Linien in der Topologie kontextuell verbessern, indem man farblich je nach Protokoll kennzeichnet. Ein Beispiel hierfür wäre, dass die deskriptive Verbindung farblich markiert werden, also der PROFIBUS einer Topologie mit Blau anzeigen und die PROFINET IRT mit Grün.

Eine andere sinnvoller umsetzbare Erweiterung für den Topologie-Editor wäre die Übertragungssicherheit was heutzutage in der industriellen Kommunikation einer großen Bedeutung hat.

Immer noch im Einsatzbereich der Inbetriebnahme eines Netzwerks aber außerhalb der Industrie, kann das Topologie-Editor als interaktives Hilfsmittel beim Lernen der verschiedenen Topologie-Arten dienen. Man könnte das Plugin um die Möglichkeit erweitern, dass ein Benutzer Die Topologie-Arten aus der Abschnitt 2.2 umschalten kann

Da Topologie-Editor vielseitig einsetzbar konzipiert wurde, sind auch die mögliche Weiterentwicklung nicht nur auf der Konfiguration beschränkt, vorstellbar ist auch der Einsatz als Werkzeug zum Planen und Projektieren mit Kunden.

Ein letzter Punkt der Verbesserung ist ganz klar die Erweiterung des Topologie-Editor um einen größeren Mehrwert zu schaffen. Die statistische Auswertung ist noch in ihren Kinderschuhen. Sie biete bisher Basisfunktionalität und muss weiter ausgebaut werden. Das beinhaltet die Erweiterung der Kriterien, der Filter und der Vergleichsmöglichkeiten.

Anhang: Use Cases

Use Case	Topologie erstellen und anzeigen
ID	**#01**
Beschreibung	Der Nutzer kann zu jeder Zeit eine Topologie erstellen sobald dass der Topologie-Editor-Server in ComStudio läuft. d.h ComStudio startet, neue Feld Busse Projekt anlegen, ein Master-Device einfügen, ein oder mehre Slave-Device einfügen. Eine Topologie anzeigen bedeutet, ein Graph in Topologie-Editor anzeigen mit den spezifischen Geräten.
Beteiligte Akteure	Inbetriebnehmer / Kunde
Vorbedingung	ComStudio läuft mit einer Projekt
Trigger	Der Nutzer klickt auf „Topologie-Editor“
Nachbedingung	
Komplexität	Hoch:
Priorität	Hoch: essentielle Funktionen
GUI	Abbildung 22

Tabelle 4: Use Case #1

Use Case	Topologie bearbeiten
ID	**#02**
Beschreibung	Der Nutzer kann eine Topologie fast zu jeder Zeit bearbeiten, d.h. den Namen eines Geräts ändern, die Verbindungen tauschen oder sogar ein neueres Gerät in der bestehenden Topologie einfügen. Die Topologie kann nur geändert werden, falls ComStudio nicht in Online-Modul steht.
Beteiligte Akteure	Inbetriebnehmer / Kunde
Vorbedingung	ComStudio und Topologie-Editor –Server läufen und ComStudio hat mindestens ein FDT-Projekt zur Verfügung
Trigger	Der Nutzer klickt auf „Topologie-Editor“

Nachbedingung	Alle Änderungen an der Daten der Topologie werden zu Topologie-Editor-Server gesendet und bewertet, um festzustellen ob die Aktion zulässig ist oder nicht.
Komplexität	Hoch: Die GUI zum Bearbeiten aller Daten der Topologie ist komplex. Auch Client-Server-Kommunikation beinhaltet komplexe Mechanismen.
Priorität	Hoch: essentielle Funktionen, da die Funktion häufig genutzt wird

Tabelle 5: Use Case #2

Use Case	Topologie Fehler erkennen/-behandeln
ID	**#03**
Beschreibung	Der Nutzer muss zu jeder Zeit die Fehler bei Nutzung der Plugin erkennen und behandeln. Beispielsweise wenn ein Kommunikationsfehler zwischen Client und Server in Form eines Netzwerkausfalls zustande kommt, soll das Plugin das Fehler erkennen und behandeln können. Bei Netzwerkausfall zeigt Topologie-Editor ein roter Hintergrund und ComStudio zeigt die textuelle dazu hörige Fehlermeldung in dem Output-Frame,
Beteiligte Akteure	Inbetriebnehmer / Kunde
Vorbedingung	ComStudio und Topologie-Editor –Server läufen und ComStudio besitzt mindestens ein FDT-Projekt zur Verfügung
Trigger	Der Nutzer klickt auf „Topologie-Editor“
Nachbedingung	Alle Änderungen an der Daten der Topologie werden behalten
Komplexität	Hoch: Die GUI zum Bearbeiten aller Daten der Topologie ist komplex. Auch Client-Server-Kommunikation beinhaltet komplexe Mechanismen.
Priorität	Hoch: Die Funktion, dass ein Nutzer datenfehlerfrei Information in das System übertragen ist essentiell für den Erfolg des Systems.

Tabelle 6: Use Case #3

Use Case	Topologie ausdrucken
ID	**#04**
Beschreibung	Direkt nach dem Erstellen einer Topologie erhält der Nutzer einen Überblick mit der wichtigsten Information zu der laufenden Topologie als Graph. Von hier kann er leicht die dargestellte Topologie ausdrucken.
Beteiligte Akteure	Kunde
Vorbedingung	Topologie ist erstellt. Bzw. aufgerufen
Trigger	Der Nutzer klickt auf „Topologie-Editor-Print“ oder navigiert zu der Tab Topologie in den Ribbon-Frame von ComStudio
Nachbedingung	Alle Änderungen an der Daten der Topologie werden behalten
Komplexität	Niedrig: Sowohl die GUI als auch die Kommunikation mit dem Server sind wenig komplex.
Priorität	Mittel: Das Ausdrucken des Topologie ist nicht essentiell für die Abläufe im System

Tabelle 7: Use Case #4

Literatur

AG, S. (01 2018). *SIEMENS*. Abgerufen am 01. 03 2018 von http://w3.siemens.com/mcms/automation/de/industrielle-kommunikation/profibus/Seiten/Default.aspx

Alexandroff, Paul, Hopf, Heinz. (1945). *Topologie.* Digitalisierungsprojekts Springer Book Archives.

cmb-systeme. (05 2004). *CMB SYSTEME*. Abgerufen am 19. 01 2018 von https://www.cmb-systeme.de/technikwissen/was-bedeutet-ethernet

Conrads, D. D. (2014). Telekommunikation. In *Grundlagen, Verfahren, Netze, Ausgabe 5* (S. 13-15). Wiesbaden: Vieweg & Sohn Verlag.

Facebook Inc. (2018). *React*. Abgerufen am 05. 05 2018 von https://reactjs.org/

Facebook Inc. (2018). *React Native*. (React) Abgerufen am 05. 05 2018 von https://facebook.github.io/react-native/

Facebook OpenSource. (2018). *React*. (Facebook Inc.) Abgerufen am 01. 03 2018 von https://reactjs.org/docs/introducing-jsx.html

Facebook/React. (2018). *Github*. Abgerufen am 05. 05 2018 von https://github.com/facebook/react/releases/tag/v16.3.2

Google. (01 2010). *Angular- Resolve*. Abgerufen am 11. 02 2018 von https://angular.io/api/router/Resolve

Google. (2018). *Angular*. (Google) Abgerufen am 05. 05 2018 von https://angular.io/guide/architecture-services

Gottfried Vossen, K.-U. W. (2016). *Grundkurs Theoretische Informatik.* Trier: Vieweg+Teubner Verlag .

Heinrich Hippenmeyer, T. M. (2016). *Automatische Identifikation für Industrie 4.0.* Berlin, Heidelberg: Springer Vieweg.

Heinrich Hübscher, H.-J. P. (2011). *IT-Handbuch.* Braunschweig: Westermann Schroedel Diesterweg Schöningh Winklers GmbH.

Hilscher Gesellschaft für Systemautomation mbH. (02 2016). *Hilscher Gesellschaft für Systemautomation mbH*. (netX) Abgerufen am 02. 02 2018 von https://www.hilscher.com/de/netx/einfuehrung/

Hilscher GmbH. (01. 10 2017). *Portal Hilscher*. Von https://kb.hilscher.com/display/COMSTUDIO/2017-06-02+Topology+in+Communication+Studio abgerufen

HMS. (2015). (HMS Industrial Networks) Abgerufen am 05. 05 2018 von http://www.feldbusse.de/Profinet/profinet.shtml

IT Wissen Info. (2001). *IT Wissen Info*. (2018) Abgerufen am 02. 02 2018 von https://www.itwissen.info/Ethernet-Ethernet.html

Kersken, S. (2013). *IT-Handbuch für Fachinformatiker.* Bonn: Galileo Press.

KUNBUS GmbH. (2016). *KUNBUS*. ( ) Abgerufen am 01. 03 2018 von https://www.kunbus.de/feldbus.html

LinkedIn. (2018). *LinkedIn*. Abgerufen am 05. 05 2018 von https://www.linkedin.com/in/evanyou/de

MDN web docs. (01 2005). *Web API Referenz*. Abgerufen am 01. 03 2018 von https://developer.mozilla.org/de/docs/Web/API

Microsoft . (02 2018). *Visual Studio-IDE*. Abgerufen am 04. 04 2018 von https://www.visualstudio.com/de/vs/

Microsoft. (01 2018). *mplementieren grundlegende CRUD-Funktionalität mit Entity Framework in ASP.NET MVC-Anwendung*. Abgerufen am 12. 04 2018 von https://docs.microsoft.com/de-de/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-basic-crud-functionality-with-the-entity-framework-in-asp-net-mvc-application

Microsoft. (01 2018). *Visual Studio Code - Code Editing*. Abgerufen am 04. 04 2018 von https://code.visualstudio.com

Microsoft Corporation. (2018). *Microsoft* . (ASP.NET Core) Abgerufen am 04 2018 von https://docs.microsoft.com/de-de/aspnet/

Northwoods Software. (01. 01 1998). (GOJS) Abgerufen am 05. 04 2018 von https://www.nwoods.com/store/p-82-gojs-oem.aspx

Northwoods Software. (01. 01 1998). *GoJS*. Abgerufen am 04. 04 2018 von https://gojs.net/latest/index.html

Olbrich, A. (2003). *Netze — Protokolle — Spezifikationen.* Wiesbaden: Vieweg & Sohn.

Plenk, P. D.-I. (2017). *Angewandte Netzwerktechnik kompakt.* Wiesbaden: Springer Fachmedien Wiesbaden GmbH.

Popp, M. (2016). *Industrielle Kommunikation mit PROFINET*. Karlsruhe: ProfiBus Nutzerorganisation e.V.(PNO).

Progress Software Corporation . (2018). *www.progress.com*. (Progress Software Corporation) Abgerufen am 05. 05 2018 von https://www.progress.com/kendo-ui

Ringhand, Z. (2010). IT- Handbuch. In *IT-Systemelektroniker/-in Fachinformatiker/-in* (S. 314-315). München: Westermann.

SAP. (01 2018). *UI5 Demo Kit - UI Development Toolkit for HTML5*. (UI5) Abgerufen am . 03 2018 von https://sapui5.hana.ondemand.com

Schnabel, P. (2018). *Elektronik-Kompendium.de - Elektronik einfach und leicht verständlich*. Von https://www.elektronik-kompendium.de/sites/net/0503281.htm abgerufen

Sommerville, I. (2007). *IT Informatik Software Engineering.* Pearson.

Twitter. (01 2010). *GetBootstrap*. (Bootstrap) Abgerufen am 02 2018 von https://getbootstrap.com

Vue.Js, G. (2018). *GitHub*. Abgerufen am 05. 05 2018 von https://github.com/vuejs/vue/releases/tag/v2.5.16

w3schools.com. (01 1999). *W3.CSS Templates*. Abgerufen am 03. 03 2018 von https://www.w3schools.com/w3css/w3css_templates.asp

Weebly. (01 2016). *netzwerk-topologien.weebly*. Abgerufen am 01. 03 2018 von https://netzwerk-topologien.weebly.com/grad.html

You, E. (2018). *VueJS*. Abgerufen am 05. 05 2018 von https://vuejs.org

Index

Glossar

API	Application Programming Interface
CANopen	Controller Area Network, the Open Communication Solution Dissemination Project
ComStudio	Communication Studio
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CSS	Cascading Style Sheets
DOM	Document Object Model
DTM	Device Type Manager
FDT	Field Device Tool
FR	Functional Requirement
GUI	Graphic User Interface
HART	Highway Addressable Remote Transducer
HTML	Hypertext Mark-up Language
IEC	International Electrotechnical Commission
IP	Internet Protocol
IRT	Isochronous Real Time
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LAN	Local Area Network
MVC	Model View Controller
MVVM	Model View Viewmodel
NFR	Non-functional Requirement
OEM	Original Equipment Manufacturer
OLE	Object Linking and Embedding
OPC UA	OLE for Process Control Unified Architecture
PC	Personal Compute
PLC	Programmable Logic Controller

PROFIBUS	Process Field Bus
PROFINET	Process Field Network
REST	Representational State Transfer
SPA	Single Page Application
UI	User Interface
XML	Extensible Mark-up Language
XSD	XML Schema Definition

Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist im Kapitel „Verantwortliche“ zu Beginn der Dokumentation aufgeführt.

Diesen Teil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Name der Mitverfasser: .....
.....

Datum

Unterschrift der Kandidatin / des Kandidaten