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, 5.6.2018

Vorwort und Danksagung

Die vorliegende Masterarbeit ist im Rahmen meiner Tätigkeit als Junior Softwareentwickler in der Firma Hilscher GmbH entstanden. Ich bin in der Abteilung „User Interface Group“ beschäftigt. Die Arbeit wurde parallel zur Entwicklung der Konfigurationssoftware Communication Studio entwickelt, da der entstehende Softwareteil später als Plugin[1] verwendet werden soll.

Trotz versuchter Fokussierung auf die wesentlichen Aspekte überschreitet diese Arbeit die ursprünglich geplante Seitenanzahl. Ein wesentlicher Grund dafür ist, dass ich dem Grundsatz der Transparenz des Vorgehens, wie er mir in meinem Fernstudium vermittelt wurde, so gut wie möglich gerecht werden wollte. Sehr viele Schritte meiner theoretischen und praktischen Arbeit habe ich daher versucht, zur intersubjektiven Nachvollziehbarkeit für den Leser ausführlich und verständlich zu dokumentieren und hoffe, dass mir dies gelungen ist.

Zunächst möchte ich mich an dieser Stelle bei all denjenigen bedanken, die mich während der Masterarbeit unterstützt und motiviert haben.

Ganz besonders gilt dieser Dank Herrn Frank Fährmann, der mich und meine Arbeit betreut hat. Nicht nur gaben Sie mir immer wieder durch kritisches Hinterfragen wertvolle Hinweise – auch Ihre kontinuierliche Motivation hat einen großen Teil zur Vollendung dieser Arbeit beigetragen. Sie haben mich dazu gebracht, über meine Grenzen hinaus zu denken. Vielen Dank für die Geduld und Mühen.

Darüber hinaus gebührt jedem mein Dank, der durch Korrekturen, Verbesserungsvorschläge und hilfreiche Diskussionen zur Qualität dieser Arbeit beigetragen hat. Gleiches gilt für meine neugeborene Tochter und meine Frau, die mir in dieser Zeit eine unersetzbare seelische und moralische Stütze waren.

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

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

Kurzfassung

Mit PROFINET IRT ist die Topologie ein wichtiger Bestandteil der Konfiguration geworden. In der aktuellen Konfigurationssoftware der Firma Hilscher ist die Eingabe oder das Erstellen der Topologie umständlich und nicht intuitiv gelöst. Mit der neuen Generation der Konfigurationssoftware soll das Erstellen einer Topologie dem Anwender erleichtert werden. Der Schwerpunkt liegt hierbei auf das Darstellen und Verändern von Topologien.

Im Rahmen dieser Arbeit soll für den Anwender eine Web-Komponente in HTML5 erstellt werden. Diese Komponente soll später als Plugin für neue Generation der Konfigurationssoftware Communication Studio dienen. Die grafische Oberfläche soll es hierbei ermöglichen, die einzelnen Funktionen der Visualisierung abstrakt und intuitiv zu nutzen. Des Weiteren soll die vom Anwender erstellte Topologie gespeichert und geladen werden können. Die Topologie soll unabhängig vom industriellen Kommunikationsnetzwerk (Protokoll) dargestellt werden.
Zu einem späteren Zeitpunkt soll es auch die Möglichkeit geben das Netzwerk zu scannen und die Topologie somit automatisch zu erstellen. Besonderes Augenmerk sollte hierbei auch auf eine einfache Erweiterbarkeit der Software gelegt werden. Im ersten Schritt der Arbeit soll anhand Protokolle PROFIBUS und PROFINET IRT mögliche Anforderungen aufgenommen und analysiert werden, sowie Formen der Topologie dargestellt werden. Auf dieser Basis soll ein Konzept zur Darstellung der Topologie entwickelt werden. Abschließend wird das Konzept in die Praxis umgesetzt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

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 logischen Darstellung des Datenflusses gibt aus auch eine physikalische Darstellung der Geräte ohne Berücksichtigung des Datenflusses. In dieser Arbeit soll nur die physikalische Darstellung, d. h. die Verknüpfung der Geräte eines industriellen Netzwerks betrachtet werden. Die logischen Verbindungen sollen in Zukunft nur eine weitere Sicht auf das Netzwerk sein. In der industriellen Kommunikation kommen nun noch verschiedene Anforderung hinzu wie z.B. Echtzeitverhalten und Übertragungssicherheit.. Grundsätzlich unterstützt die Firma Hilscher alle gängigen Netzwerkprotokolle. In den vergangenen Jahren ist eine Vielzahl neuer und Protokolle hinzugekommen. Die Anforderungen an Netzwerke, die aus der Industrie kommen, steigen stetig. Durch die permanent steigenden Anforderungen steigt auch der Aufwand in der Konfiguration, der neu entstehenden Netzwerkeprotokolle. 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 aber nicht unwichtiger Grund ist die Reduzierung des Support-Aufkommens, der sich aus einer selbsterklärenden Software ergibt. Zur Konfiguration eines Netzwerkes gehören noch weitere feldbusprotokollabhängige Parameter und Einstellungen, die hier an dieser Stelle nicht betrachtet werden und als gegeben vorausgesetzt werden wie z.B. die Übertragungsgeschwindigkeit der Daten. Die grafischen Darstellungen sind ein ideales Mittel für die Visualisierung von komplexeren Topologien. Durch die Visualisierung der Topologie des Netzwerkes kann dem Anwender ein schnellerer und leichterer Überblick sowie Einstieg in das Netzwerk gegeben werden. 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 Software, das einheitliche „Look and Feel“ sowie das gleiche Bedienkonzept der Software für den Kunden weitere Vorteile.

1.1 Grund

Der Grund für diese Arbeit ist, das die Hilscher Gesellschaft für Systemautomation eine Möglichkeit sucht, komplexe industrielle Netzwerke grafisch darzustellen und zu konfigurieren. Die Ausgangssituation ist, dass es momentan keine Möglichkeit gibt, eine physikalische Topologie

eines Netzwerkes unabhängig vom Netzwerkprotokoll zu erstellen. Die entstehende Komponente soll später als Plugin für die bereits bestehende Software Communication Studio der Firma Hilscher integriert werden. Wie in der Einleitung erwähnt, gibt es verschiedene Gründe eine unabhängige Topologie-Komponente zu erstellen. Neben der Wiederverwendbarkeit und auch der Kostenreduktion ist meiner Meinung nach einer der wichtigsten Gründe diese Komponente als grafische Oberfläche zu entwerfen. Der Grund ist, dass der Mensch diese Art der visuellen Informationen am schnellsten Erfassen und Verstehen kann. Da ich auch selbst großes Interesse habe, grafische Oberflächen zu entwerfen und zu entwickeln, habe ich mich für diese Masterarbeit entschieden.

1.2 Ziele

Eines der Ziele, die mit dieser Masterarbeit erreicht werden sollen, ist, dass die Kunde der Firma Hilscher einfach und mit wenig Aufwand seine Aufgaben, das Netzwerk zu konfigurieren, erfüllen kann. Hierzu soll sich der Anwender nicht erst das entsprechende Fachwissen aneignen müssen. Wie auch schon in der Einleitung erwähnt, sollte es das Ziel sein, das Support-Aufkommen zu reduzieren. Andere Ziele wie Wiederverwendbarkeit oder Integration in eine bestehende Software spielen natürlich auch eine wichtige Rolle.
Im bisherigen Entwicklungsfokus der Firma Hilscher standen zunächst nur die Kernfunktion zur Konfiguration eines industriellen Netzwerkes der neuen Software Communication Studio im Vordergrund. Durch die gezielte Integration eines Software Plugin-Moduls „*Topology Editor*“[2] in die Software Communication Studio können positive auf Modellierung eines Netzwerkes einwirken. Dabei stellt sich die Frage, wie das Plugin für die bestehende Software entwickelt werden soll, insbesondere unter der Voraussetzung der oben genannten Ziele sowie auch geringer struktureller Veränderungen der bestehenden Software.
Meine persönlichen Ziele, die ich mit dieser Masterarbeit erreichen möchte, sollen ein tieferes und umfassendes Wissen über die Netzwerktopologie in der industrielleren Kommunikation sein. Weiterhin soll mir die Implementierung dieser Software die Aufgaben und Pflichten der Rolle eines Senior Softwareentwickler in einem Unternehmen aufzeigen. Im Rahmen dieser Arbeit möchte ich mir fundierte Fachkenntnisse aneignen und so die Fähigkeit erlangen, komplexe Vorhaben zu planen und umzusetzen.
Aus diesem Grund soll im Rahmen der vorliegenden Masterarbeit eine Möglichkeit zur Integration des Plugin in die Konfigurations-Software Communication Studio entstehen.

1.3 Aufgaben

Durch die Ziele ergeben sich nun folgende Aufgaben im Rahmen dieser Masterarbeit. Zum einen sollen verschiedenen Software-Komponenten entstehen, die unabhängig vom Netzwerkprotokoll die Topologie des industriellen Kommunikationsnetzwerkes darstellen und zum anderen auch die Bearbeitung in einem gewissen Rahmen zulassen.
Da dieses Themengebiet sehr komplex ist und über viel Fachwissen voraussetzt, erfolgt im ersten Schritt eine Einführung und Einarbeitung in das Thema der Arbeit, bevor anhand einer Analyse der Protokolle PROFIBUS und PROFINET IRT Formen der Topologie und verschiedene Darstellungsarten analysiert werden. Auf Basis der durch die Evaluierung gewonnenen Erkenntnisse wird im zweiten Schritt ein Konzept zur Darstellung entwickelt. Nach der Erstellung des Konzeptes erfolgt dann die Implementierung. Die Implementierung erfolgt dann in mehreren Iterationsstufen.

[2] ist der Name der Plugin zu entwickeln

Da die Arbeit nur über einen bestimmten zeitlichen Rahmen verfügt und in dieser Zeit prototypenhaft entwickelt werden soll, habe ich mir im Vorfeld Gedanken über eine zeitliche Einteilung gemacht, um nicht die Übersicht und das Ziel der Fertigstellung des Projektes zu verlieren. Ich habe die Masterarbeit in verschieden Phasen eingeteilt und mit einem Start und Ende versehen.

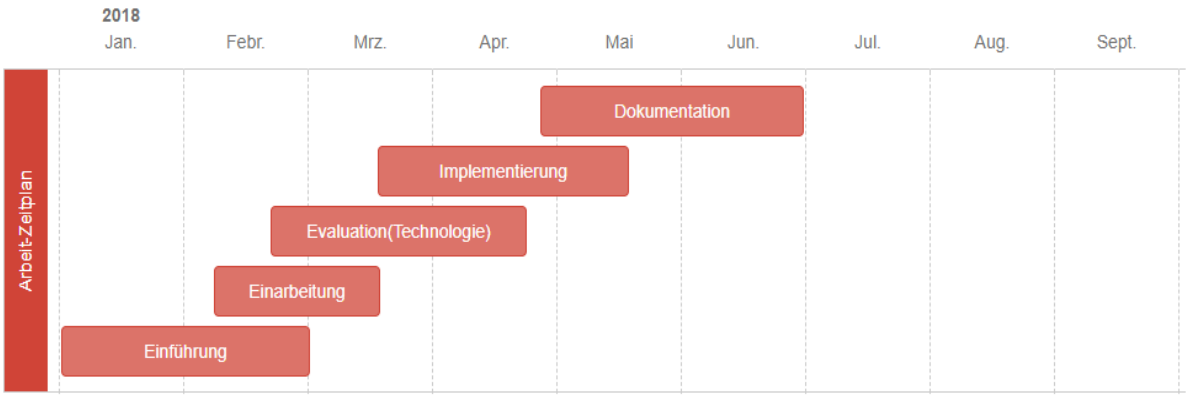


Abbildung 1: Zeitplan für die Entwicklung von Plugin Topology-Editor

Abbildung 1 zeigt Beginn und Ende der einzelnen festgelegten Projektphasen. Die Festlegung erfolgt aufgrund des festgelegten Fertigstellungstermin. Neben den Projektterminen wurden zu diesem Zeitpunkt auch die wichtigen Meilensteine festgelegt. Tabelle 1 zeigt beides: die Termin- und Meilensteinplanung.

Phase	Beschreibung	Ressource	Meilenstein
Einführung	Erklärungen und Erläuterungen der Aufgaben	Projektleiter gemeinsam mit allen Beteiligten	Ziele der Arbeit verstehen
Einarbeitung	Erarbeitung der Grundlagen und Theorie für den Fachbereich	Student (ich)	Topologien kennenlernen, Protokolle einarbeiten
Evaluation	Erstellung einer Konzeption	Student mit Gruppenleiter	Konzept erstellen
Realisierung	Erstellung des Prototyp	Student Entwicklungsumgebung VS 2017	Prototype erstellen
Dokumentation	Erstellen von Text und Grafiken	Student Word, Drucker	Schriftlichen Teil Masterarbeit erstellen

Tabelle 1: Termin- und Meilensteinplanung für die Masterarbeit

1.4 Firmenvorstellung

Die Firma Hilscher Gesellschaft für Systemautomation mbH besteht seit 1986 und beschäftigt heute über 260 Mitarbeiter an 10 Standorten weltweit. Mit der Philosophie des kontinuierlichen Wachstums aus eigener Kraft, ist das Unternehmen ein verlässlicher Partner für seine Kunden. Alle Prozesse sind nach ISO 9001 und 14001 zertifiziert.

Die Kernkompetenz von der Firma Hilscher ist die Technologie, Entwicklung und Produktion von industriellen Kommunikationslösungen für die moderne Fabrikautomation. Die Produkte reichen von PC-Karten und Gateways über OEM-Aufsteckmodule bis hin zu leistungsfähigen ASICs mit den dazugehörigen Protokoll-Stacks. Diese werden weltweit zur Kommunikation zwischen Automatisierungsgeräten und Steuerungen eingesetzt. Im Bereich PC-Karten ist die Firma Marktführer. Ein derart umfassendes Lösungs-Portfolio für Feldbusse und Real-Time-Ethernet ist das Alleinstellungsmerkmal der Firma Hilscher. Hilscher bietet auch ASICs basierte die netX-Technologie für Gerätehersteller an, inklusive Entwicklungs-Dienstleistungen und kundenspezifischer Baugruppen-Fertigung. In diesem Bereich ist die Firma nicht nur anerkannter Systempartner der großen Hersteller, sondern zählt auch eine Vielzahl von Ingenieurbüros und Systemintegratoren zu seinen Kunden. Außerdem ist Hilscher in allen Feldbus- und Real-Time-Ethernet-Organisationen vertreten

Abbildung 1: Haupteingang Hilscher Gesellschaft für Systemautomation mbH

2 Grundlagen

Dieses Kapitel ist für das Verständnis der Arbeit wichtig. Hier werden die Grundlagen erklärt. Die folgenden 3 Unterkapitel beschäftigen sich zunächst mit den allgemeinen Definitionen der Fachbegriffe in der industrielleren Kommunikation. Des Weiteren folgt ein Überblick über die Topologie-Arten und eine kurze Einführung in die Netzwerkprotokolle der industriellen Kommunikation. Hierbei werden die Protokolle PROFINET und PROFIBUS näher betrachtet.

2.1 Grundlegende Begrifflichkeiten

Bevor wir beginnen, hier die Definitionen der meist genutzten Terminologien in der Kommunikation.

- **Netzwerk**.. Als *Netzwerk* bezeichnet man den Verbund mehrerer Geräte oder Gerätegruppen zum Zweck der Datenkommunikation. Sie können nach der räumlichen Ausdehnung (Größe) bzw. durch die Art der Leitungsführung oder nach Übertragungsgeschwindigkeit unterschieden werden.
- **Ethernet-Technik:** Unter *Ethernet* versteht man eine Datenübertragungstechnik, die es ermöglicht Daten innerhalb geschlossener Netzwerke zwischen verschiedenen Geräte zu übertragen. Es umfasst die Festlegung für spezielle genormte Kabel und Steckverbindungen sowie für Paketformat und Protokolle. Ethernet ist in der Norm IEE802.3 standarisiert worden, und ist zur weitverbreitetsten LAN-Technologie geworden. Ethernet bildet die Basis für Übertragungsprotokolle wie z.B. TCP IP, PROFINET oder EtherCAT, um nur die bekanntesten zu nennen.
- **Netzwerkprotokoll:** Damit Kommunikationspartner in einem Netzwerk überhaupt miteinander kommunizieren können, müssen diese bestimmte Vereinbarungen und Regeln einhalten. Diese Regeln sind die *Protokolle*. Ein einzelnes Protokoll arbeitet immer einen bestimmten Funktionsbereich ab, der bei der Datenkommunikation der Funktionalität einer Schicht des OSI-Referenzmodell entsprechen kann.
- **OSI-Referenzmodell:** Das *OSI-Referenzmodell* [1] der Internationalen Standardisierungs-Organisation (ISO) wurde ab 1977 als Grundlage für die Bildung von Kommunikationsstandards entworfen. Ziel der Open Systems Interconnection (OSI) ist die Kommunikation in heterogenen Netzen.
- **Feldbus**: Ein *Feldbus* verbindet in einer Anlage verschiedene Sensoren (Messfühler) und Aktoren (Stellglieder) zwecks Kommunikation mit einem übergeordneten Steuerungsgerät (SPS). Dabei senden und empfangen in der Regel mehrere Teilnehmer nahezu gleichzeitig über ein und dieselbe Leitung. Damit es zu keiner Datenkollisionen kommt, muss also festgelegt werden wer (Kennung) was (Messwert, Befehl) wann (Initiative) kommuniziert.

- **Knoten & Kanten**: Knoten repräsentieren die Kommunikationspartner und die Kanten die Verbindung in der Graph Theorie.
- **Kennwerte**: Sind die Eigenschaften einer Topologie.
- **Topologie**: Teilgebiet der Mathematik, welches ursprünglich diejenigen Eigenschaften geometrischer Gebilde behandelt, die bei umkehrbar eindeutigen, stetigen und totalen Abbildungen erhalten bleiben, das sogenannte Homomorphismus [2]. In der industrielleren Kommunikation beschreibt die Topologie die Verbindungen der teilnehmenden Geräte untereinander, die den Datenverkehr ermöglichen.
- **Controller**: 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 [3].
- **Supervisor**: Dies kann ein Programmiergerät, Personal Computer oder Human-Man-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.

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 Vor- und Nachteilen der Topologie kann man auch schon erkennen für welches Einsatzgebiet diese Systeme verwendet werden können.
Topologien werden grafisch (nach der Graphentheorie) 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 Kennwerte einer Netzwerktopologie

Durchmesser
Der Durchmesser einer Topologie beschreibt die maximale direkte Entfernung zwischen zwei Knoten in Hops[3]. Damit ist er ein direktes Maß für die zu erwartenden maximalen Transferzeiten, d.h. je größer der Durchmesser, desto größer die Transferzeit im ungünstigsten Fall.

Grad
Der Grad einer Topologie gibt die Anzahl der Links pro Knoten an. Diese kann für jeden Knoten gleich oder verschieden sein. Haben alle Knoten einer Topologie den gleichen Grad, so ist

[3] nennt man in Netzwerk den Weg von einem Netzknoten zum nächsten

die Topologie regulär, was sich vorteilhaft auf das Netzwerk auswirkt. Außerdem beschreibt der Grad indirekt, welche Kosten man zum Aufbau der Topologie aufbringen muss. Je höher der Grad, desto höher sind die Kosten.

Bisektionsweite
Die Bisektionsweite gibt die minimale Anzahl von Links an, die durchschritten werden müssen, um ein Netz mit N Knoten in zwei Netze mit jeweils N/2 Knoten zu teilen. Damit ist sie ein Maß für die Leistungsfähigkeit eines Netzes, da in vielen Algorithmen die Knoten der einen Netzhälfte mit den Knoten der anderen Hälfte kommunizieren. Je niedriger also die Bisektionsweite, desto ungünstiger wirkt sich dies auf den Zeitbedarf für den Datenaustausch zwischen beiden Netzhälften aus.

Symmetrie
Bei einer symmetrischen Topologie sieht das Netz von jedem Betrachtungspunkt (Knoten/Links) gleich aus, d.h.es existieren für Knoten und/oder Kanten sogenannte Automorphismen d.h. einfach gesprochen, dass sich Knoten und/oder Links in einem symmetrischen Netz gleich verhalten, egal welchen Knoten oder welchen Link man betrachtet. Dies hat äußerst positive Auswirkungen (nämlich eine Vereinfachung) auf die Programmierung, die Lastverteilung und das Routing, da es keine Spezialfälle zu betrachten gibt.

Skalierbarkeit
Die Skalierbarkeit gibt das kleinste Netzwerkinkrement (Anzahl von Knoten und Links) an, um das man eine Topologie erweitern muss, um keine Leistungseinbußen und die Beibehaltung topologietypischer Eigenschaften nach der Erweiterung zu garantieren.

Konnektivität
Die Konnektivität gibt die minimale Anzahl von Knoten oder Links (Kanten- bzw. Knotenkonnektivität) an, die durchtrennt werden müssen, damit das Netz als solches nicht mehr funktionstüchtig ist. Sie ist ein Maß für die Anzahl der unabhängigen Wege, die es zwischen zwei verschiedenen Knoten geben kann. Damit beschreibt sie auch die Ausfallsicherheit des Netzes, d.h. je höher die Konnektivität, desto ausfallsicherer ist das Netz.

Physische Topologie: beschreibt den Aufbau der Netzverkabelung

Logische Topologie: beschreibt den Datenfluss zwischen den Endgeräten d.h. in welcher logischen Beziehung die Geräte beim Datenaustausch zueinanderstehen.

2.2.2 Linien-Topologie

Die Linien-Topologie war einer der ersten Netze, die in kleineren Firmen und privaten Haushalten zu finden war. Ein Koaxialkabel[4] bildet die Basis, die Linie, welches beliebig, bis zu einen gewissen Grad, verlängert werden kann. Die Stationen werden über T-Stücke an das Kabel angeschlossen. Am Anfang und Ende des Kabels sind Widerstände angebracht um Reflexionen[5] zu verhindern.

[4] Kupferkabel mit einem konzentrischen, elektrischen Leiter, der durch eine Isolierung in einem festen, mittigen Abstand zur Abschirmung gehalten wird
[5] *Reflektionen* sind Wellen, die sich in beiden Richtungen auf der Übertragungsleitung fortpflanzen. So wie ein Echo.

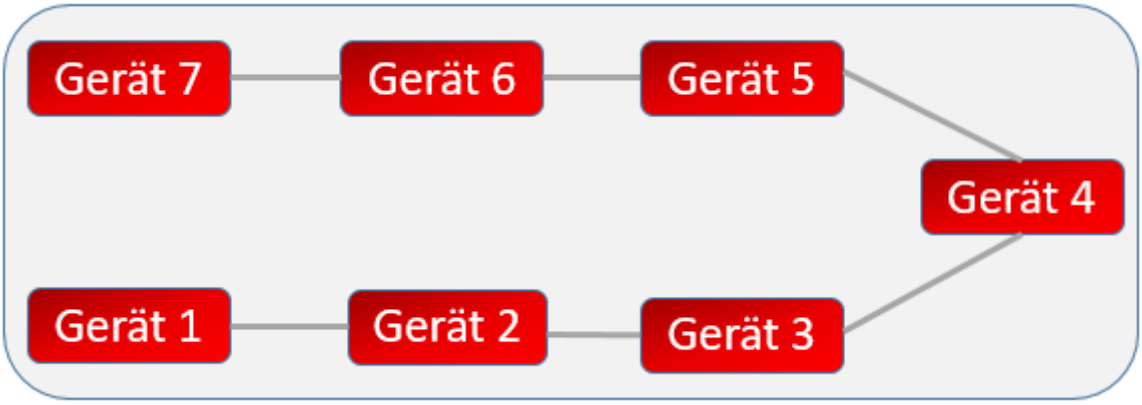


Abbildung 2: Linien-Topologie

Vorteile

- Nur geringe Kosten, da nur geringe Kabelmengen erforderlich sind
- Einfache Verkabelung und Netzerweiterung
- Es werden keine weiteren Geräte zur Übermittlung der Daten benötigt

Nachteile

- Alle Daten werden über ein einziges Kabel übertragen
- Datenübertragungen können leicht abgehört werden (Sicherheitsproblem)
- Eine Störung des Übertragungsmediums an einer einzigen Stelle im Netz (defektes Kabel) blockiert den gesamten Netzwerkstrang
- Es kann immer nur eine Station Daten senden. Während der Sendung sind alle anderen Stationen blockiert (Datenstau)
- Aufgrund der Möglichkeit der Kollisionen sollte das Medium nur zu ca. 30% ausgelastet werden

2.2.3 Ring-Topologie

Eine Ring-Topologie ist eigentlich eine Erweiterung der Linien-Topologie, d.h. das erste und das letzte Gerät sind miteinander verbunden. Abbildung 3: Ring-Topologie zeigt eine Ring-Topologie im Einsatz, dieses System ist sehr einfach aufzubauen.

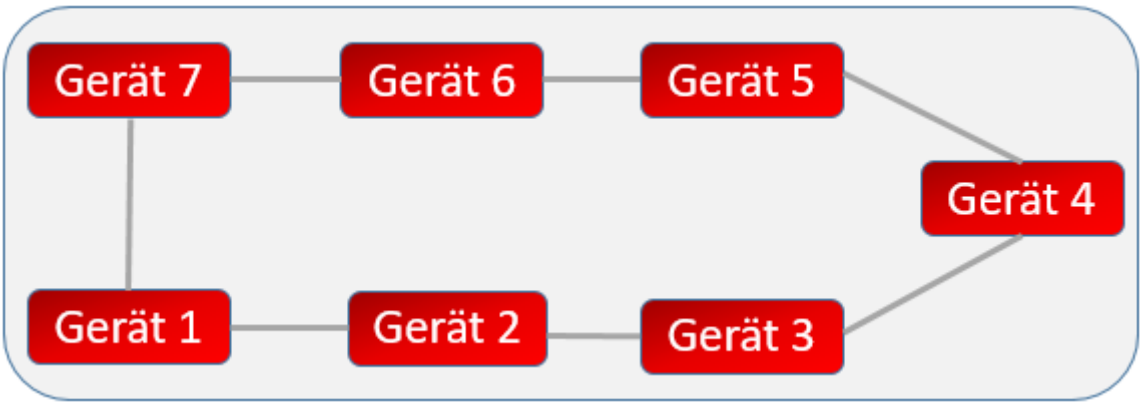


Abbildung 3: Ring-Topologie

Vorteile

- Deterministische Netzwerkkommunikation - Vorgänger und Nachfolger sind definiert
- Alle Teilnehmer haben gleiche Zugriffsmöglichkeiten
- Garantierte Übertragungsbandbreite
- Sehr gut skalierbar, der Grad bleibt bei Erweiterung konstant
- Reguläre Topologie, daher leicht programmierbar

Nachteile

- Teure Komponenten
- Relativ hoher Durchmesser
- Höherer Verkabelungsaufwand

2.2.4 Stern-Topologie

Charakteristisches Merkmal der Stern-Topologie sind kurze Wege, das bedeutet, dass zwischen Sender und Empfänger nur wenige Vermittlungsstationen vorhanden sind. Abbildung 4 zeigt eine klassische Sterntopologie, diese besteht aus einem zentralen Element (d. h. Switch) und 4 Geräte, die mit einem Ende-zu-Ende Verbindung angeschlossen sind. Fällt ein Gerät aus, stört es die Kommunikation die übrigen Geräte nicht, solange diese nicht mit dem ausgefallenen Gerät kommunizieren. Fäll der Switch aus, so kann kein Gerät mehr Daten austauschen. Dies ist die einzige Schwachstelle dieser Topologie.

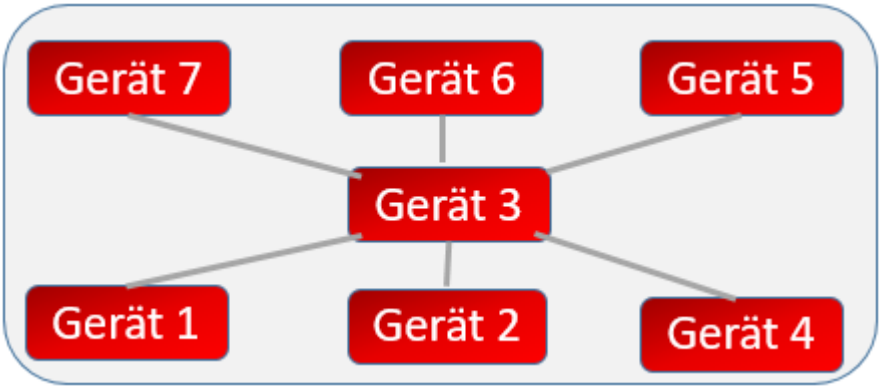


Abbildung 4: Stern-Topologie

Vorteile

- Der Ausfall eines Endgerätes hat keine Auswirkung auf den Rest des Netzes.
- Dieses Netzwerk bietet hohe Übertragungsraten, wenn der Netzknoten ein Switch ist.
- Leicht erweiterbar
- Leicht verständlich
- Leichte Fehlersuche
- Sehr gute Eignung für Multicast-/Broadcastanwendungen
- Kein Routing benötigt

Nachteile

- Durch Ausfall des Verteilers (Switch) wird Netzverkehr unmöglich
- Niedrige Übertragungsrate bei vielen Geräten.

2.2.5 Baum-Topologie

Eine Erweiterung zum Stern stellt der Baum dar (siehe Abbildung 5). Derzeit ist dies die meist genutzte Netzwerktopologie. Prinzipiell wäre eine Verkabelung von jedem Switch zu jedem andern Switch möglich.

Vorteile

- Der Ausfall eines Endgeräts hat keine Konsequenzen
- Strukturelle Erweiterbarkeit
- Gute Eignung für Such- und Sortieralgorithmen

Nachteile

- Bei Ausfall eines Verteilers (Wurzel) ist der ganze davon ausgehende Zweig des Verteilers nicht mehr erreichbar.
- Zur Wurzel hin kann es bedingt durch die für Bäume definierte Bisektionsweite von 1 zu Engpässen kommen, da zur Kommunikation von der einen unteren Baumhälfte in die andere Hälfte immer über die Wurzel gegangen werden muss.
- Bäume haben mit zunehmender Tiefe (=Anzahl der Links von der Wurzel bis zu einem Blatt) einen sehr hohen Durchmesser. Dies führt in Verbindung mit der Bisektionsweite (größer als 1) zu schlechten Latenzeigenschaften bei klassischen Bäumen.

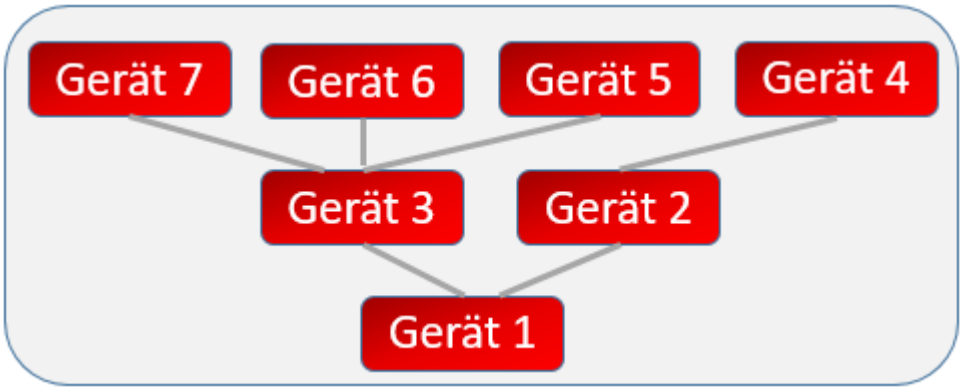


Abbildung 5: Baum-Topologie

2.2.6 Vermaschtes Netz

Es gibt zwei Arten von vermaschten Topologien. In der ersten, der sogenannten vollständig vermaschten Topologie, weist jeder Knoten eine direkte Verbindung zu jedem anderen Knoten auf. In einer teilweise vermaschten Topologie sind Knoten jeweils nur mit den Knoten verbunden, mit denen sie am meisten interagieren.

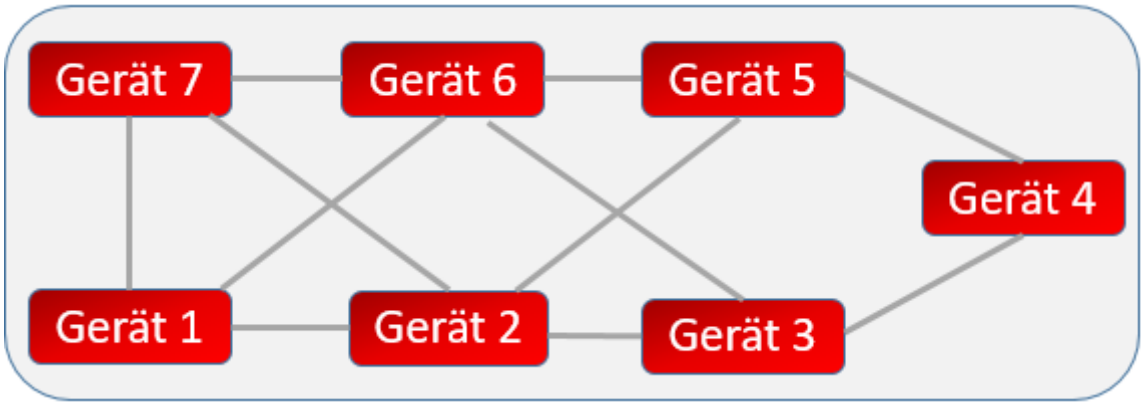


Abbildung 6: Vermaschtes Netzwerk

Nachteile:

- Viele Kabel notwendig
- Sehr hoher Energieverbrauch
- Vergleichsweise komplexes Routing nötig

Vorteile:

- Sicherste Variante eines Netzwerkes
- Bei Ausfall eins Endgerätes ist durch Umleitung die Datenkommunikation weiterhin möglich
- Sehr leistungsfähig

2.3 Netzwerke-Protokolle

In der industriellen Kommunikation gibt es viele verschiedene Netzwerkprotokolle, die zum Einsatz kommen, beispielsweise PROFIBUS, CANopen, DeviceNet, Sercos, 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 **Fehler! Verweisquelle konnte nicht gefunden werden.** wird kurz erklärt was der PROFIBUS kann und kurz auf charakteristischen Merkmale der PROFIBUS-Geräte eingegangen. Das Protokoll PROFINET wird dann im anschließenden Abschnitt **Fehler! Verweisquelle konnte nicht gefunden werden.** betrachtet.

2.3.1 PROFIBUS

PROFIBUS war einst ein von SIEMENS [5] 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 Gerätetypen:

- **Master**: Er ist ein sogenannter „aktiver Teilnehmer“, der den gesamten Datenverkehr auf dem Bus bestimmt. Er ist als einziger befugt, unaufgefordert auf eine Ressource zuzugreifen und Nachrichten ohne externe Aufforderung zu versenden.

- **Slave:** Der Slave ist im Gegenzug „passive Teilnehmer“, welche keine eigenständige Zugriffsberechtigung auf die Ressource habt. 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, 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

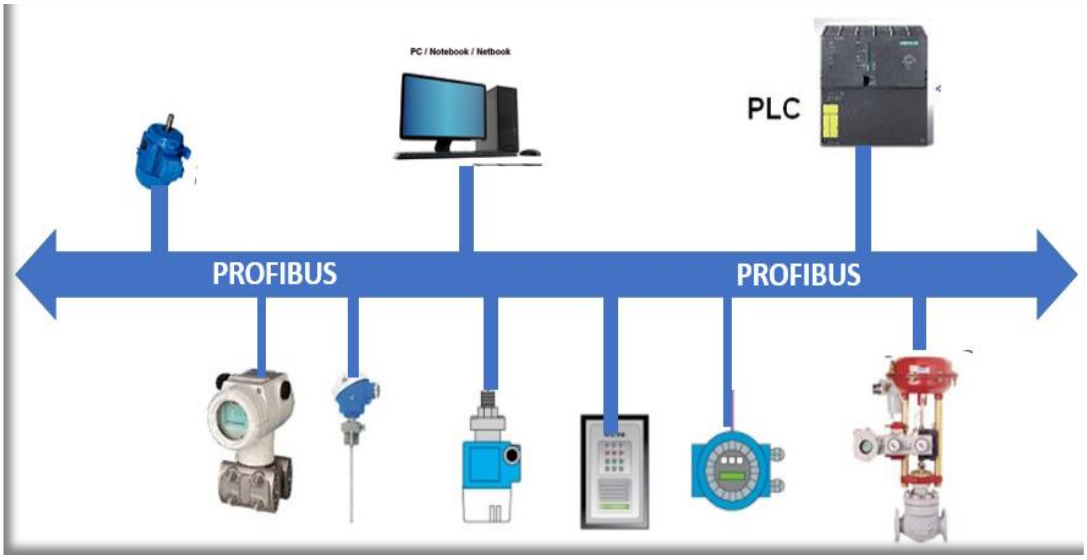


Abbildung 7: Beispielhafte PROFIBUS-Topologie an einem Master-Slave-System

Abbildung 7 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.

2.3.3

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 auch Daten, die für den Datenaustausch mit dem Device von Bedeutung sind. Die Nutzerorganisation schreibt zwingend die Lieferung einer GSD-Datei vor. Diese ist Bestandteil des Zertifizierungstests.

3 Ausgangssituation (Istzustand)

Die Hilscher Gesellschaft für Systemautomation mbH hat die Fachkompetenz für die Entwicklung und Produktion industrieller Kommunikationslösungen für die moderne Fabrikautomation. Mit der Tendenz „Industrie 4.0“ [7] 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 web-basierte 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 Techniken 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 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. Grund für die Ablösung der Software durch das Communication Studio 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# verwendet und die Oberflächen wurden mit WPF 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.
Fehler! Verweisquelle konnte nicht gefunden werden. zeigt die wichtigsten Module des „Communication Studios“. Diese Module bieten den Vorteil, das die Software mit den steigenden Anforderungen mitwachsen kann.

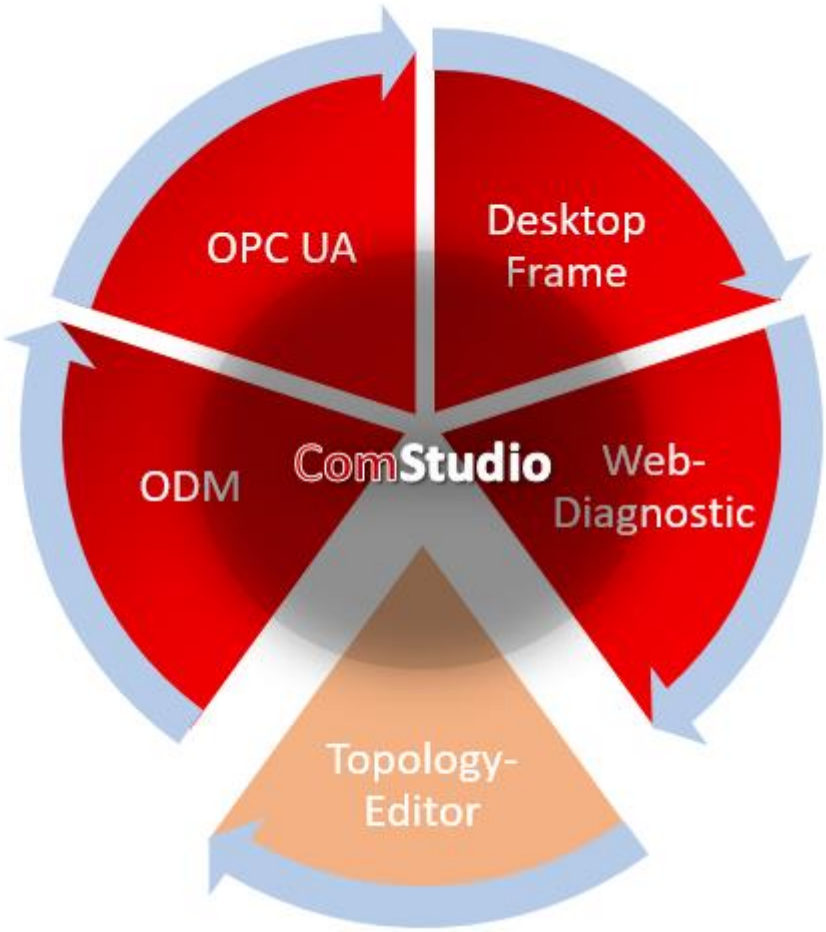


Abbildung 8: Vereinfachtes Übersichtschema *ComStudio* mit zukünftiger Topology-Editor

3.2 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 Communication Studio 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.

3.3 Auswertung

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

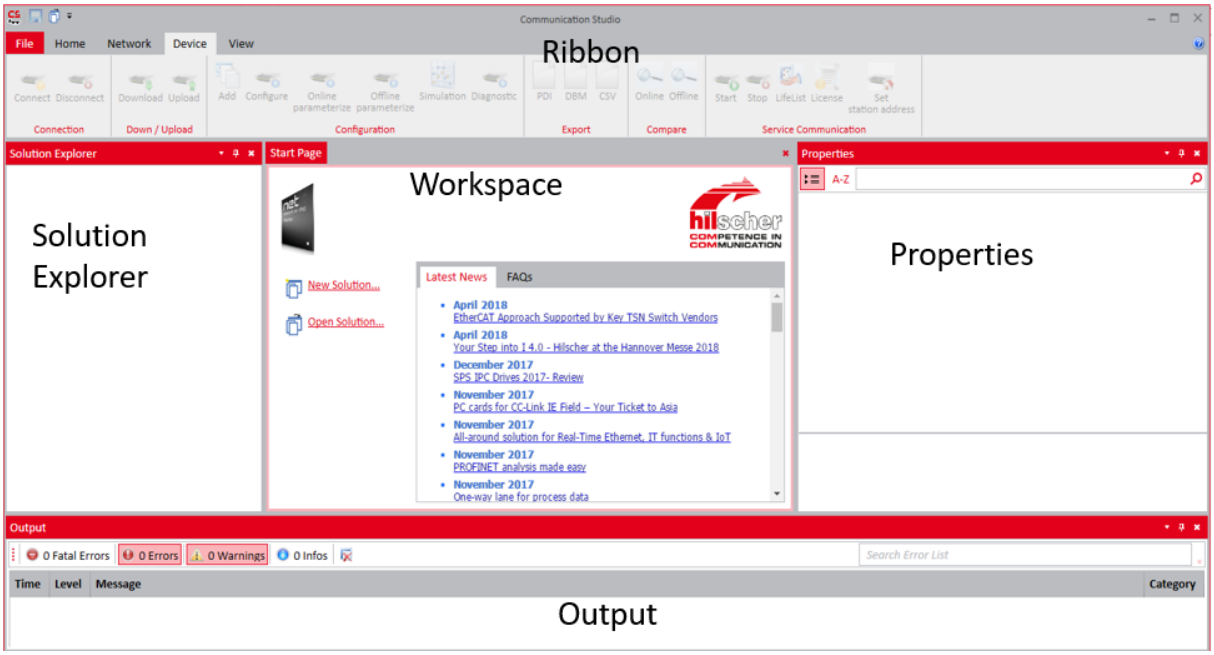


Abbildung 9: Benutzeroberfläche von Communication Studio mit spezifische Ansichten: Der Nutzer sieht die Liste der FDT-Projekte welche in eine Solution zusammengefasst sind, detaillierte Information spezifische Gerätetype bzw. Verbindungstypen, Ergebnisse in bereits geschlossen Prozesse und kann in Workspace eine neue GUI öffnen(z.B. DTM-GUI)

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 ~~in~~ sogenannten „*Solution Explorer*“ (Abbildung 6) 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 9)
- 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 [8].
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[6]) 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-Editor. Weiterhin formuliert man die Kann-Kriterien, die umgesetzt werden, wenn es die Ressourcen und die Planung des Projektes zulassen und ~~die~~ 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.

[6] FR - 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[7]), 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 10 gibt bereits einen Überblick über einige der Anforderungen.
Prinzipiell lassen sich bei der Interaktion mit dem System zweier Nutzerrollen identifizieren. Bei Anzeigen und Erstellen der Netzwerktopologie sind hier die Kunde und Inbetriebnehmer zu nennen.
Die Kunde bzw. Inbetriebnehmer der Firma Hilscher möchten im System eine Topologie anzeigen. Sie sollen durch Communication Studio den entsprechende FDT-Projekt anlegen oder

[7] *NFR* - Nonfunctional requirement

importiert (falls schon vorhanden ist) die DTMs laden. Nach erfolgreicher Einfügen der DTMs, kann der Kunde Die Topologie anzeigen. Somit können Sie hier ohne Rechte die entsprechende Netzwerktopologie bearbeiten.

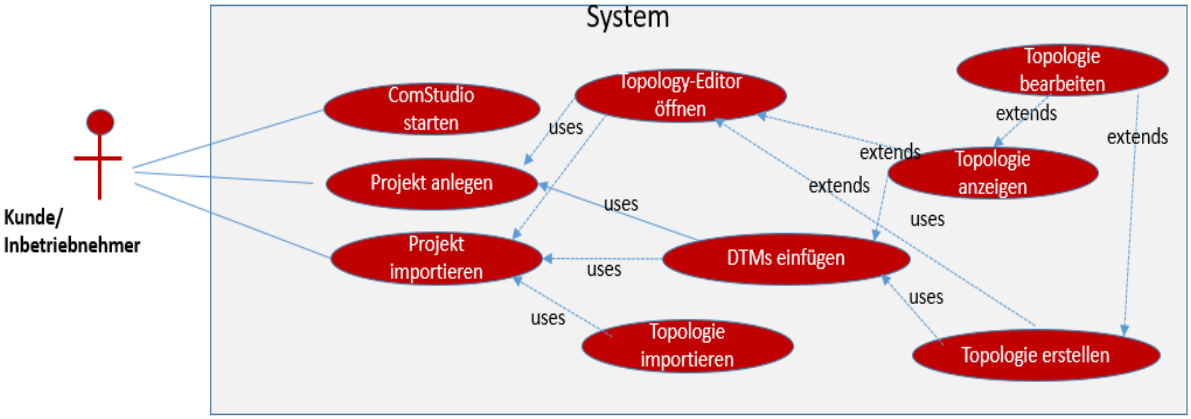


Abbildung 10: Anwendungsfall-Diagramm: Ausgewählte Funktionen des Systems

Während der Bearbeitung haben die Anwender prinzipiell zwei Optionen vorzugehen:

- **Hinzuzufügen eines neuen DTM**: Communication Studio bietet schon ein Verfahren um ein DTM in System hinzufügen. Der Nutzer sollten 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 allererste das DTM aus Topology-Editor selektieren, dadurch wird die DELETE-Funktion aktiviert, die im Ribbon von Communication Studio liegt.

5 Evaluation der Technologien

TopologyEditor 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 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 den 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 Umsetzung mit vergleichbare Aufstellung als überblick über den gefundenen Rahmenwerken. Für die serverseitige Umsetzung wird die ASP.NET.Core 2.0 eingesetzt. Anschließend führen wir das Tools bzw. die externen Bibliotheken zur Vereinfachung und Beschleunigung der Entwicklung, zum Beispiel für die Visualisierung von Information Daten wir die Bibliothek GoJS genutzt und für das Styling und die Strukturierung von Benutzeroberfläche wird das Bootstrap 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 Kommunikation und Konfiguration, die mit einem modularen System einhergehen, an. In diesem Abschnitt werden nur 3 ausgewählte Konzepte vorstellt. 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 Angular [9]

Das Entwickeln und Erstellen von HTML-Seiten, 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

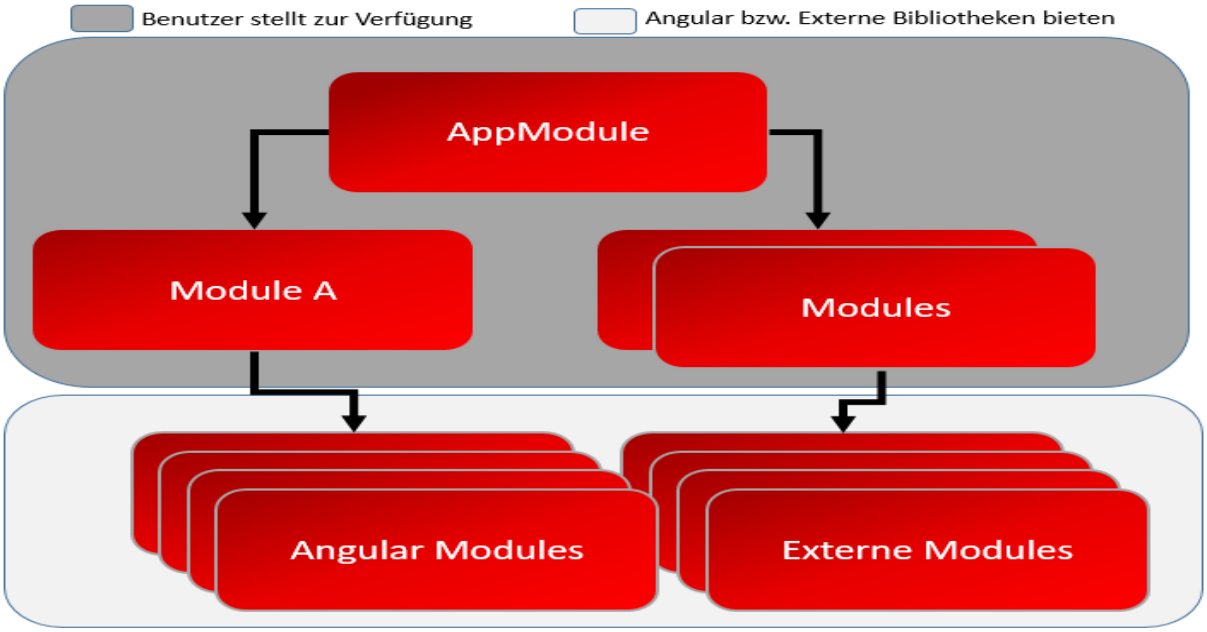


Abbildung 11: Typische zusammengesetzte Anwendungsarchitektur mit dem Angular Framework

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

- **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 vorgesehen, sondern eine Trennung nach MVC-Muster. Der Entwickler definiert dafür ein Tag 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 (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 Details 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 [9]

5.1.2 Vue.js [10]

Vue ist ein fortschrittliches Framework für den Aufbau von Benutzeroberflächen, das von Evan You [11] im Jahr 2013 entwickelt wurde. Die Aktuelle Version ist 2.5.16 [12]. 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 einem Ökosystem von unterstützenden Bibliotheken, welche sie bei der Bewältigung der Komplexität großer Einseitenanwendungen unterstützt. Dieses Framework arbeitet nach dem Entwurfsmuster MVVM und ist stark von Angular inspiriert.

5.1.3 SAPUI5 [13]

SAPUI5 ist eine JavaScript-Framework, die von der Firma SAP 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-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.
- Es biete integrierte Unterstützung für Architekturkonzept wie Zwei-Wege-Datenbindung und Routing
- Es biete die Bindung mit JSON, XML und andere Datenformaten

5.1.4 React [14]

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 Unterstutzern. Die Aktuelle Version 16.3.2 [15] 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 [16]. Darüber hinaus kann es auch auf dem Server diese UIs erstellen und in mobile Apps mit *React Native* [17] ausführen.

5.1.5 Kendo UI [18]

Das bulgarische Consulting-Unternehmen Telerik bietet die Kendo UI-Framework für das mobile und web Einsatzfeld an. Der Code läuft nicht in einer WebView[8], sonders 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[9]-Struktur.
Eine gute gemachte Kendo-UI Applikation unterscheidet sich daher nicht wesentlich von einer komplett nativen und lässt sich zudem automatisch zwischen Betriebssystem portieren.
Der Kendo UI ist ein der finanzstarke Konzern, der – anders als der React-Geldgeber Facebook- im Großem und Ganzen ausschließlich von Dienstleistung für Entwickler lebt.

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

Es gibt das Desktop-Programm Sidekick[10], das die einst angebotene Cloud-Kompilation abstrahiert und Projektskelette generiert

5.1.6 GoJS [19]

GoJS 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.

Jedes Diagramm besteht aus Knoten, die durch Links verbunden sein könnten und die optional in Gruppen gruppiert 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)

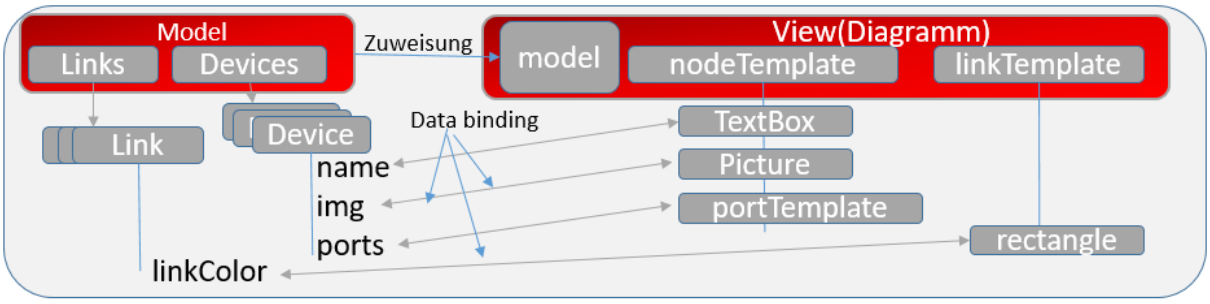


Abbildung 12: Die wichtigsten Eigenschaften des Diagramms und ihre Interaktion

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.

[10] Mehr Info unter https://github.com/cloudsidekick

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, z. B. das Auswählen von Teilen oder das Ziehen von Elementen oder das Zeichnen einer neuen Verknüpfung zwischen zwei Knoten. Der ToolManager [11]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 Rückgängigmachen / 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 GraphObjects 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 erscheint 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.

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

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 [20] 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 einschließlich betrachtet die Programmierkonzepte.

5.2.1 Lernkurve und Kosten

In der industrielleren 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 Meisterarbeit in einer zeitlichen Periode begrenzt ist, sollte diese leicht und schneller 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 zueinander stellen. 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** Bei der Anwendungsentwicklung sollte dem Entwickler möglichst die Wahl gelassen werden, in welcher Programmiersprache

entwickelt wird. Eine domain-spezifische Frage zielt dabei auf die Möglichkeit der grafischen Programmierung. Als Programmiersprachen sind JavaScript, C# und TypeScript denkbar.

- **Unterstützungsbibliotheken** Vordefinierte Komponenten z.B. für Verhaltensauswahl, Sicherheit, Monitoring und Internationalisierung erleichtern den Entwicklungsprozess und fördern die Wiederverwendung von Software-Modulen, wobei gegebenenfalls entsprechende Anpassungen erforderlich sind.
- **Lizenzbedingung** Der Typ der Lizenz der Frameworks bestimmt insbesondere im Fall der kommerziellen Nutzung über deren generelle Anwendbarkeit. Durch das gewählte Lizenzmodell wird die Breite der Entwicklungs-Community zumindest mitbestimmt. Eine aktive Community erleichtert die Entwicklungsarbeit und bietet in Wikis oder Foren 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 X zeigt eine Übersicht der clientseitigen Frameworks bezüglich der jeweiligen Vergleichskriterien:

Für die Umsetzung des TopologyEditor wird das Framework Angular 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 diesem Framework ist nur in geringen Maße notwendig, da vorhandenes Wissen aus dem Bereich der Webtechnologie angewendet werden kann, ohne eine Konzepte 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

Clientseitige Frameworks						
	React	**Angular**	**Vue**	**GoJS**	**SAPUI5**	**KendoUI**
Lernkurve	mittel	leicht	mittel	leicht	mittel	schwer
Kostenlose Variante	+(einge-schränkt)	+	+(einge-schränkt)	-	-	-
Kommuni-kationsprin-zip	+	+	+	k.A.	+	+
Kommuni-kations-middleware	+	+	+	k.A.	+	+
Sprachen	TypeScript	Type-Script	JavaScript	Ja-vaScript	JavaScript	JavaScript, C#
Eigene IDE	-	+	-	-	-	-
Open-Source-Framework	+	+	+(teil-weise)	+(teil-weise)	-	+(teil-weise)
Lizenz	+	+	+	+	+	+
Preis	k.A.	k.A.	k.A.	ab 2995 US-$ pro Jahre	k.A.	ab 899 US-$ pro Jahr
+ vorhanden	**- nicht vorhanden**			**k.A. keine Angabe**		

Tabelle 2: kurze Überblick über die jeweils wichtigsten Kriterien der beschriebenen Frameworks und damit über ihre jeweilige Einsatzmöglichkeit.

5.4 Bootstrap [20]

Bootstrap ist ein OpenSource 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 (Integration von JQuery)

Alle Elemente sind gestaltet, wodurch sichergestellt wird, dass jede der möglichen Komponenten einheitlich aussieht

Das Framework ist rückwärtskompatibel. So wird sichergestellt, dass die einzelnen Elemente der Website auf allen Browsern dargestellt werden können.

Bootstrap ist anpassbar oder 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

5.5 ASP.NET Core [21]

Bei „*TopolgyEditor*“ 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 sind, müssen die eingesetzten Technologien Plattformunabhängig und unabhängig von einer bestimmten Programmiersprache sein. REST[12] ist eine Möglichkeit, um Web Services zu implementieren. Da wir die C#-Programmiersprache gewählt haben, werden wir als Framework ASP.NET Core 2.0 von Microsoft nehmen.

5.5.1 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[13] .

Ein REST-System besteht aus Resources, die per URI[14] 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.

[12] Steht für Representational State Transfer

[13] Steht für Simple Object Access Protocol: Protokoll zum Austausch strukturierter Informationen

[14] Uniform Resource Identifier

- Jede Anfrage muss alle notwendigen Informationen für die Durchführung beinhalten (da http *stateless* ist)
- Darstellungen werden untereinander verlinkt, um dem Client die Möglichkeit zu geben, von einem Zustand in den nächsten zu wechseln.
- Weder Client noch Server müssen die Bedeutung einer URI verstehen
- 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

5.5.2 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
- Fähigkeit zur Erstellung und Ausführung unter Windows, macOS und Linux
- Open Source und mit Fokus auf der Community
- ASP.NET Core besteht vollständig aus NuGet-Paketen.

6 Grundprinzipien der Topology-Editor

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 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.

6.2 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.

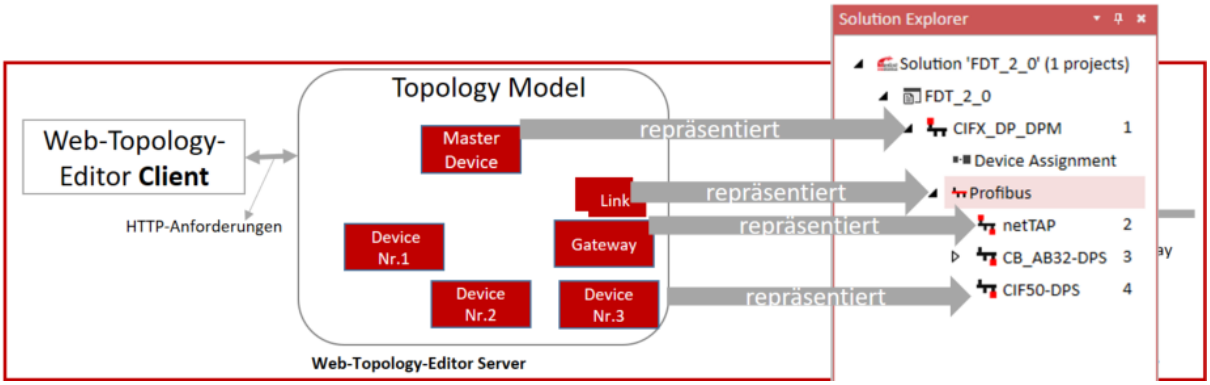
Der Hauptanwendungsfall für Topology-Editor ist der Offline/Online-Datenaustausch zwischen Geräten und browserfähige Dokumente mit Data-Access-Funktionalität um die Topologie des entsprechenden Systems darzustellen. In diesem Anwendungsfall werden die Gerätedaten von einem Topology-Editor-Server bereitgestellt und von einem Topology-Editor-Client konsumiert, der in das browserfähige Dokument integriert ist.
Topology-Editor bietet Funktionen zum Darstellen eines physikalischen Anordnung (Verknüpfungen falls es zulässt) der protokollabhängige 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.

6.3 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.

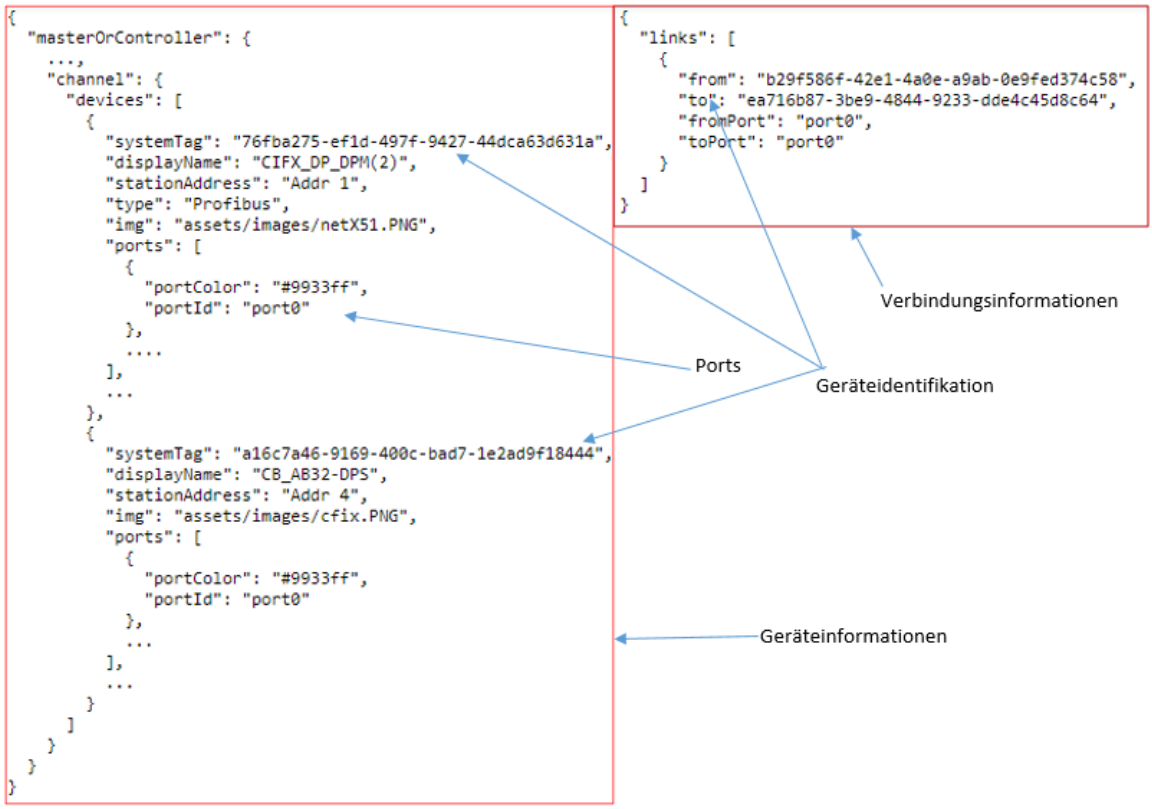


Abbildung 13: Die Output JSON-Dateien für die Mini-Topologie

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[15] 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.

[15] Single Page Application

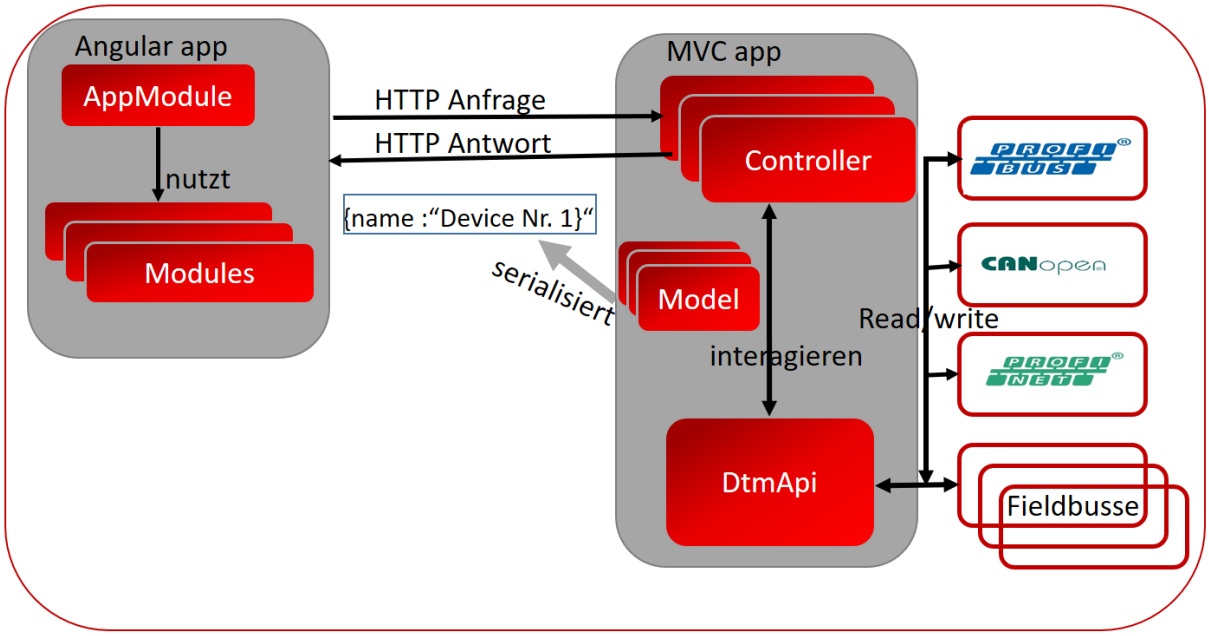


Abbildung 14: Diagramm zeigt den Grundlegenden Entwurf der Topology-Editor

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 15). Eine vollständige Übersicht befindet sich im angehängte CD.
Aus den funktionalen Anforderungen ergibt sich direkt das Domänenmodell.

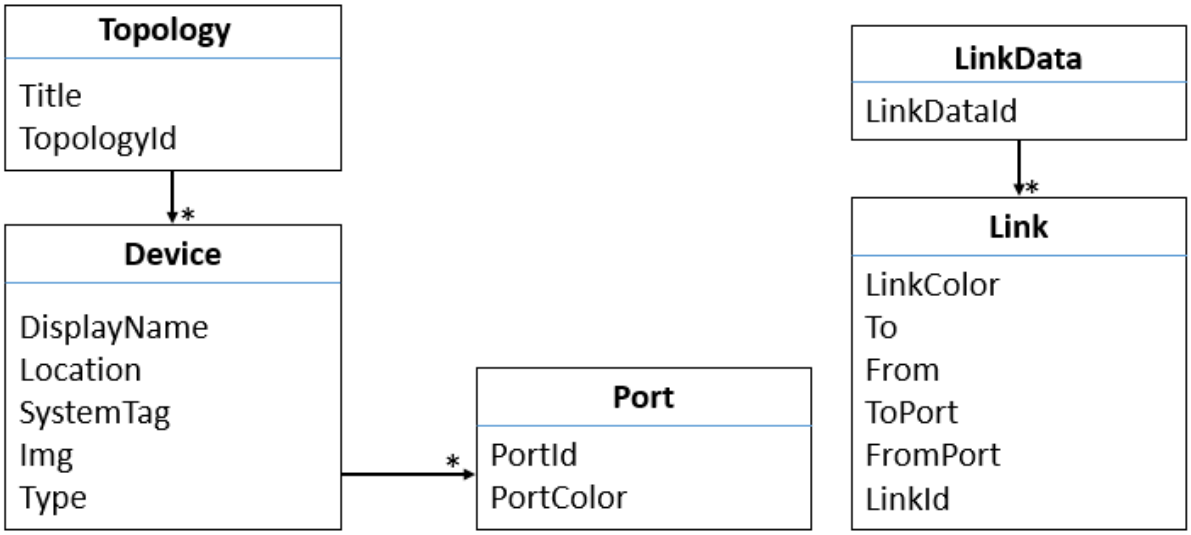


Abbildung 15: Domänenmodell der Elemente des Topology-Editor

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.

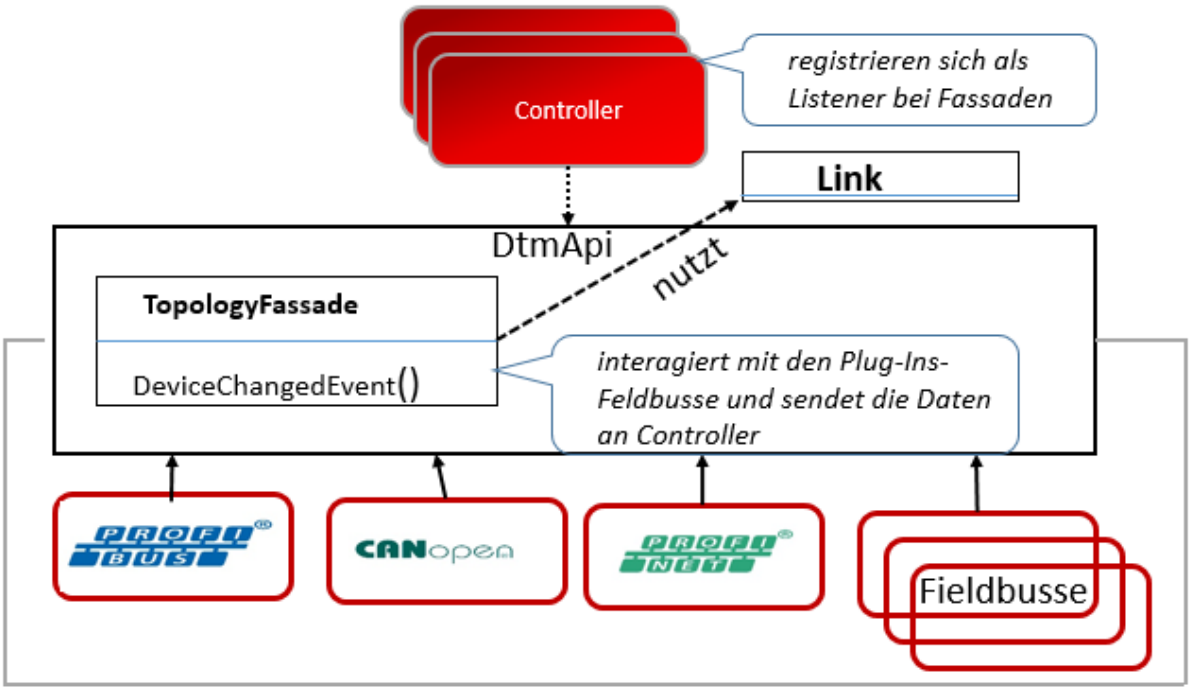


Abbildung 16: Klassendiagramm eines Beispiel für Fassaden mit feldbusabhängigen Protokollen

Abbildung 16 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 Gesellschaft ist das *Scrum*[16]-Vorgehensmodell in der Abteilung 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 „*Web-TopologyEditor“* wurde als Haupterkzeug für die serverseitige Umsetzung die Entwicklungsumgebung Microsoft Visual Studio 2017 mit der Programmiersprache C# gewählt, für clientseitige Umsetzung der Visual Studio Code. Des Weiteren werden Bootstrap und ASP:NET Core verwendet, wie bereits in Kapitel 6 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. Im Kapitel 3 wird betrachtet, wie mit

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

Hilfe der gefundenen Daten die Topologie erstellt wird. Im Folgenden Kapitel 4 steht die Darstellung der fehlerbehafteten Statistik im Vordergrund. Abschließend wird in Kapitel 5 die grafische Umsetzung der Darstellung und die Auswertung erläutert.

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

Die TopologyEditor-Library unterstützt die Software Communication Studio dabei, alle Netzwerk-Protokolle für Web-Services mithilfe des Model-View-Controller-Musters zu kommunizieren. Diese Library unterstützt diejenigen, die mit ASP.NET Core eine Reihe von Anwendungsstilen erstellen möchten. Sie wurde 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.

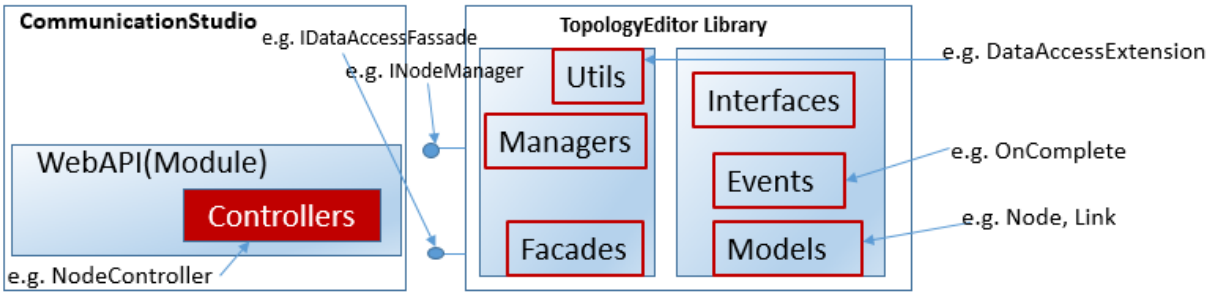


Abbildung 17: Zusammengesetzte Anwendungsarchitektur mit häufig verwendeten Paketen

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 dasErstellen von Modellkassen erleichtern sollen,und definiert statische schreibgeschützte Eigenschaften, um spezifische Parameter bei Codierung zugänglich zu machen.

Hilscher.TopologyEditor.Web.API ist eine Sammlung von Controllern, die daie 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,2 die zum Teil nach der Evaluation der JavaScript-Frameworks (siehe Abschnitt 12) 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-TopologyEditor selbst zu geben und aufzuzeigen, wie sich diese Einflüsse außen, wird das Modul-Diagramm mit Klassen von Client-TopologyEditor besprochen und dabei eine Übersicht über die Rolle von JSON-Dateien als Informationsquelle von TopologyEditor gegeben.

Zur Veranschaulichung der eigens für Client-TopologyEditor und der aus Angular-Framework und GoJs-Bibliothek verwendeten Klassen wurde das Diagramm in Abbildung 111 erstellt. Es zeigt das AppModule in der Mitte, um das sich alle Klassen von Client Topology Editor anordnen. Als Einstieg in das Diagramm können alle Komponente-Klassen und Service-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 Edito, 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 ein 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.

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-Operators [10] umgesetzt.

Weitere Funktionalität, die auch über das Beenden der Plugin hinaus agieren kann, wird unter Angular als Angular Resolver[10] 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 konsomiert
- „LinkService.ts“: liefert die Liste der Verbindungen, die zur gewählten Topologie gehört.

Für die Kommunikation mit den beiden JSON-Dateien (siehe Abschnitt 2) 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 [10] 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 Communication Studio 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.

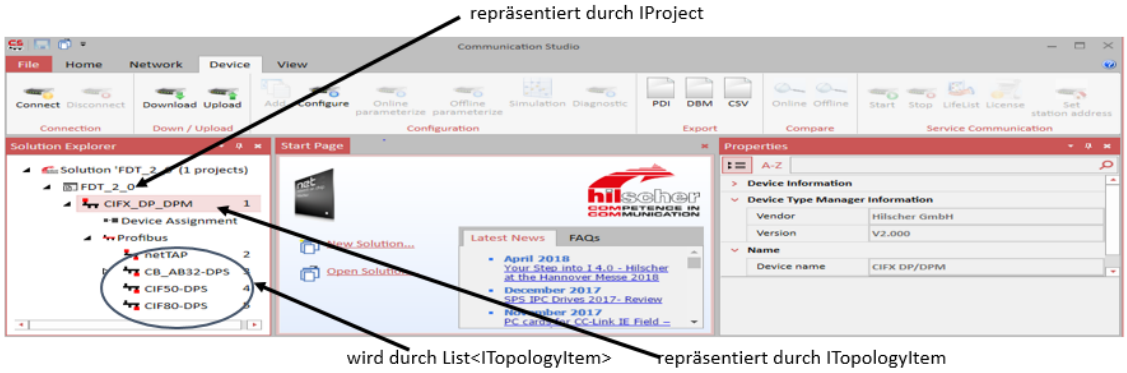


Abbildung 18: Erstellen eines FDT-Projekt

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.

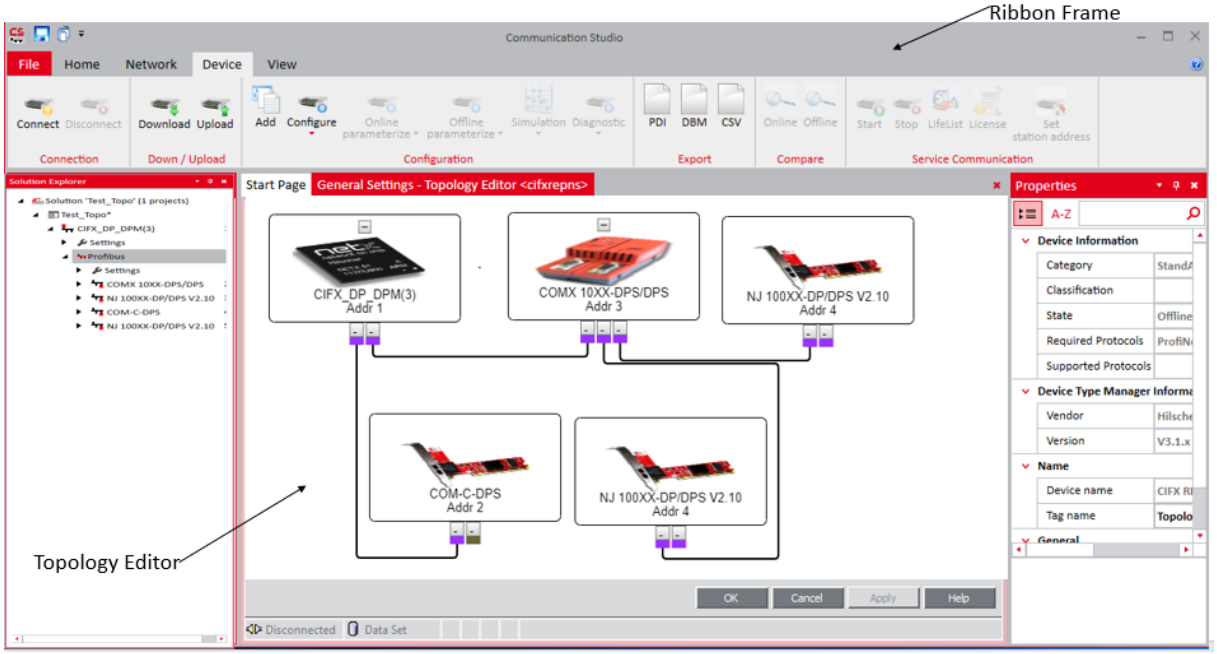


Abbildung 19: Communication Studio mit Topology Editor im Einsatz: Der Benutzer muss im Editor zum Erstellen und Anpassen einer Topologie je nach Anzahl der Devices scrollen, um alle Information einsehen und bearbeiten zu können. In der Abbildung ist die detaillierte Information über ein Device bzw. 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 19

Tabelle 3: 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 4: 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 5: 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 6: Use Case #4

Literatur

[1] S. Kersken, IT-Handbuch für Fachinformatiker, Bonn: Galileo Press, 2013.

[2] K.-U. W. Gottfried Vossen, Grundkurs Theoretische Informatik, Trier: Vieweg+Teubner Verlag , 2016.

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

[4] K. GMBH, „KUNBUS," , 01 2016. [Online]. Available: https://www.kunbus.de/CANopen-grundlagen.html. [Zugriff am 01 03 2018].

[5] S. AG, „SIEMENS," 01 2018. [Online]. Available: http://w3.siemens.com/mcms/automation/de/industrielle-kommunikation/profibus/Seiten/Default.aspx. [Zugriff am 01 03 2018].

[6] HMS, HMS Industrial Networks, 2015. [Online]. Available: http://www.feldbusse.de/Profinet/profinet.shtml. [Zugriff am 05 05 2018].

[7] T. M. Heinrich Hippenmeyer, Automatische Identifikation für Industrie 4.0, Berlin, Heidelberg: Springer Vieweg, 2016.

[8] I. Sommerville, IT Informatik Software Engineering, Pearson, 2007.

[9] Google, „Angular," Google , 2018. [Online]. Available: https://angular.io/guide/architecture-services. [Zugriff am 05 05 2018].

[10] G. Vue.Js, „GitHub," 2018. [Online]. Available: https://github.com/vuejs/vue/releases/tag/v2.5.16. [Zugriff am 05 05 2018].

[11] LinkedIn, „LinkedIn," 2018. [Online]. Available: https://www.linkedin.com/in/evanyou/de. [Zugriff am 05 05 2018].

[12] SAP, „UI5 Demo Kit - UI Development Toolkit for HTML5," UI5, 01 2018. [Online]. Available: https://sapui5.hana.ondemand.com. [Zugriff am 03 2018].

[13] Facebook OpenSource, „React," Facebook Inc., 2018. [Online]. Available: https://reactjs.org/docs/introducing-jsx.html. [Zugriff am 01 03 2018].

[14] Facebook/React, „Github," 2018. [Online]. Available: https://github.com/facebook/react/releases/tag/v16.3.2. [Zugriff am 05 05 2018].

[15] Facebook Inc., „React," 2018. [Online]. Available: https://reactjs.org/. [Zugriff am 05 05 2018].

[16] Facebook Inc., „React Native," React, 2018. [Online]. Available: https://facebook.github.io/react-native/. [Zugriff am 05 05 2018].

[17] Progress Software Corporation , „www.progress.com,“ Progress Software Corporation , 2018. [Online]. Available: https://www.progress.com/kendo-ui. [Zugriff am 05 05 2018].

[18] Northwoods Software, „GoJS,“ 01 01 1998. [Online]. Available: https://gojs.net/latest/index.html. [Zugriff am 04 04 2018].

[19] „Northwoods Software,“ GOJS, 01 01 1998. [Online]. Available: https://www.nwoods.com/store/p-82-gojs-oem.aspx. [Zugriff am 05 04 2018].

[20] Twitter, „GetBootstrap,“ Bootstrap, 01 2010. [Online]. Available: https://getbootstrap.com. [Zugriff am 02 2018].

[21] Microsoft Corporation, „Microsoft,“ ASP.NET Core, 2018. [Online]. Available: https://docs.microsoft.com/de-de/aspnet/. [Zugriff am 04 2018].

[22] Hilscher GmbH, „Portal Hilscher,“ 01 10 2017. [Online]. Available: https://kb.hilscher.com/display/COMSTUDIO/2017-06-02+Topology+in+Communication+Studio.

[23] E. You, „VueJS,“ 2018. [Online]. Available: https://vuejs.org. [Zugriff am 05 05 2018].

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