

Darstellung der Topologie eines industriellen Kommunikationsnetzwerks

Master-Abschlussarbeit

Betreuer: Prof. Dr. Georg Rock und Frank Fährmann

Hattersheim, 13.7.2018

Vorwort und Danksagung

Die vorliegende Masterarbeit ist im Rahmen meiner Tätigkeit als Junior Softwareentwickler in der Firma Hilscher Gesellschaft für Systemautomation mbH (im Folgenden Firma Hilscher genannt) entstanden. Ich war in der Abteilung Software User Interface beschäftigt. Die meiner Thesis zugrundeliegende Software wurde in Zusammenarbeit mit dem Entwicklerteam des Engineering Tools Communication Studio entwickelt, da sie im letzteren als Plugin¹ verwendet werden soll.

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

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

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

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

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

Kurzfassung

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

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

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

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

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung und Problemstellung | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Zielsetzung | 2 |
| 1.3 | Aufgaben | 2 |
| 1.4 | Firmenvorstellung | 4 |
| 2 | Grundlagen | 6 |
| 2.1 | Grundlegende Begriffe | 6 |
| 2.2 | Grundlegende Netzwerktopologien | 7 |
| 2.2.1 | Linien-Topologie | 8 |
| 2.2.2 | Ring-Topologie | 9 |
| 2.2.3 | Stern-Topologie | 10 |
| 2.2.4 | Baum-Topologie | 11 |
| 2.2.5 | Vermaschtes Netz | 12 |
| 2.3 | Netzwerke-Protokolle | 12 |
| 2.3.1 | PROFIBUS | 12 |
| 2.3.2 | PROFINET | 15 |
| 3 | Ausgangssituation | 16 |
| 3.1 | FDT | 16 |
| 3.2 | Was ist Communication Studio? | 17 |
| 3.3 | Befragungsmethoden | 18 |
| 3.4 | Auswertung | 18 |
| 4 | Anforderungsanalyse | 20 |
| 4.1 | Funktionale Anforderungen | 20 |
| 4.1.1 | Muss-Kriterien | 20 |
| 4.1.2 | Kann-Kriterien | 21 |
| 4.2 | Nichtfunktionale Anforderungen | 22 |
| 4.2.1 | Muss-Kriterien | 22 |
| 4.2.2 | Kann-Kriterien | 22 |
| 4.3 | Anwendungsfälle | 22 |
| 5 | Evaluation der Technologien | 24 |
| 5.1 | Clientseitig relevante Frameworks | 24 |
| 5.1.1 | JavaScript Frameworks | 25 |
| 5.1.2 | UI-JavaScript-Bibliotheken | 26 |
| 5.2 | Vergleichskriterien | 28 |

| | | |
|----------|---|-----------|
| 5.2.1 | Lernkurve und Kosten | 29 |
| 5.2.2 | Kommunikationsmechanismen | 29 |
| 5.2.3 | Programmierkonzept | 29 |
| 5.3 | Ergebnisse der Analyse | 30 |
| 5.4 | Bootstrap | 31 |
| 5.5 | ASP.NET Core | 32 |
| 6 | Spezifikation und Konzept | 34 |
| 6.1 | Topology-Editor, grundlegende Konzepte | 34 |
| 6.2 | ComStudio-Szenario | 35 |
| 6.3 | ComStudio-Szenario, User Interface | 36 |
| 6.4 | Schnittstellen zwischen Komponenten | 36 |
| 6.5 | Datendarstellung (Prozessdaten, Bedienungsdaten) | 37 |
| 6.6 | Zukunftsaspekte (evtl. Erweiterungen) | 40 |
| 6.7 | Entwurf | 41 |
| 6.7.1 | Architektur des Topology-Editor | 41 |
| 6.7.2 | Entwurf der Serverkomponente | 42 |
| 6.7.3 | Datenmodell | 43 |
| 6.7.4 | DtmApi Wrapper | 44 |
| 7 | Realisierung | 45 |
| 7.1 | Vorgehensweise | 45 |
| 7.2 | Verwendete Werkzeuge | 45 |
| 7.3 | Ausgewählte Implementierungsaspekte | 45 |
| 7.3.1 | Realisierung der Server-Applikation | 46 |
| 7.3.2 | Realisierung der Client-Applikation | 47 |
| 7.3.3 | Systemvorbereitung und Erstellen eines FDT-Projekts | 48 |
| 7.3.4 | Anzeigen einer Topologie | 49 |
| 8 | Zusammenfassung und Ausblick | 51 |
| 8.1 | Zusammenfassung | 51 |
| 8.2 | Ausblick | 52 |
| | Literatur | 57 |
| | Index | 59 |
| | Glossar | 60 |
| | Erklärung der Kandidatin / des Kandidaten | 62 |

Abbildungsverzeichnis

| | |
|--|----|
| Abbildung 1: Zeitplan für die Entwicklung von Plugin Topology-Editor | 3 |
| Abbildung 2: Haupteingang Hilscher Gesellschaft für Systemautomation mbH | 5 |
| Abbildung 3: Linien-Topologie | 8 |
| Abbildung 4: Ring-Topologie | 9 |
| Abbildung 5: Stern-Topologie | 10 |
| Abbildung 6: Baum-Topologie | 11 |
| Abbildung 7: Vermashtes Netzwerk | 12 |
| Abbildung 8: PROFIBUS basiert auf international etablierte Standard | 13 |
| Abbildung 9: Beispielhafte PROFIBUS-Topologie an einem Master-Slave-System..... | 14 |
| Abbildung 10: Vereinfachtes Übersichtsdiagramm ComStudio mit Topology-Editor | 17 |
| Abbildung 11: Benutzeroberfläche von ComStudio mit spezifischen Ansichten..... | 18 |
| Abbildung 12: Anwendungsfall-Diagramm: Ausgewählte Funktionen des Systems | 23 |
| Abbildung 13: ComStudio mit dem Topology-Editor Komponenten..... | 34 |
| Abbildung 14: Kommunikation zwischen Feldbussystemen, ComStudio und Topology- Editor-Komponente | 35 |
| Abbildung 15: Das Layout für die Benutzeroberflächenkomponente | 36 |
| Abbildung 16: Topology-Editor Schnittstellen | 37 |
| Abbildung 17: Datenhandhabungskonzept für Topology-Editor | 38 |
| Abbildung 18: Die Output JSON-Dateien für die Mini-Topologie | 39 |
| Abbildung 19: Diagramm zeigt den Grundlegenden Entwurf der Topology-Editor | 42 |
| Abbildung 20: Komponentenübersicht für Topology-Editor-Server | 43 |
| Abbildung 21: Domänenmodell der Elemente des Topology-Editor..... | 43 |
| Abbildung 22: Klassendiagramm eines Beispiel für Fassaden mit feldbusabhängigen Protokollen | 44 |
| Abbildung 23: Zusammengesetzte Anwendungsarchitektur mit häufig verwendeten Paketen | 46 |
| Abbildung 24: Vereinfachtes Diagramm für die Umsetzung der clientseitigen Anwendung.. | 47 |
| Abbildung 25: Erstellen eines FDT-Projekt..... | 49 |

| | |
|---|----|
| Abbildung 26: ComStudio mit Topology Editor im Einsatz..... | 50 |
|---|----|

Tabellenverzeichnis

| | |
|--|----|
| Tabelle 1: Meilensteinplanung für die Masterarbeit | 4 |
| Tabelle 2: Vergleichstabelle für JavaScript-Framework..... | 30 |
| Tabelle 3: Vergleichstabelle für UI-Bibliotheken | 31 |
| Tabelle 4: Use Case #1 | 53 |
| Tabelle 5: Use Case #2..... | 54 |
| Tabelle 6: Use Case #3..... | 55 |
| Tabelle 7: Use Case #4..... | 56 |

1 Einleitung und Problemstellung

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

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

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

1.1 Motivation

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

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

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

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

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

1.2 Zielsetzung

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

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

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

1.3 Aufgaben

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

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

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

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

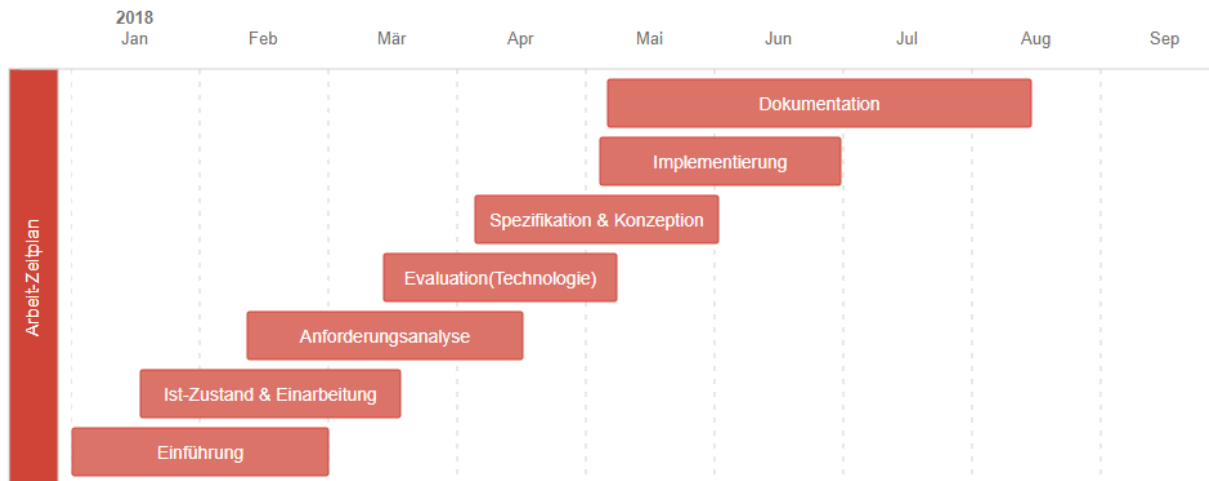


Abbildung 1: Zeitplan für die Entwicklung von Plugin Topology-Editor

In der nachfolgenden Tabelle sind diese Projektphasen in Detail aufgeschlüsselt.

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

Tabelle 1: Meilensteinplanung für die Masterarbeit

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

1.4 Firmenvorstellung

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

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

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

- PC²-Karten
- Gateways
- OEM³-Aufsteckmodule für die Automatisierungs- und Visualisierungslösungen
- ASICs basierte netX-Technologie (Hilscher Gesellschaft für Systemautomation mbH, 2016)
- Protokollstacks
- Softwarelösungen für Konfiguration, Inbetriebnahme und Diagnose, Analyse der Netzwerke
- Industrie 4.0 / IIoT⁴-Lösungen (z.B.: Edge-Gateways mit Cloudanbindungslösungen)

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



Abbildung 2: Haupteingang Hilscher Gesellschaft für Systemautomation mbH

² PC steht für Personal Computer

³ OEM steht für Original Equipment Manufacturer

⁴ IIoT steht für Industrial Internet of Thing

2 Grundlagen

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

2.1 Grundlegende Begriffe

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

- **Netzwerk** (Plenk, 2017, S. 3) - als Netzwerk bezeichnet man den Verbund mehrerer Geräte oder Gerätegruppen zum Zweck der Datenkommunikation. Die wesentlichen Bestandteile der meisten Netzwerke sind Server und Clients. Das größte Netzwerk ist heutzutage das Internet.
- **TCP⁵** (Heinrich Hübscher, 2011, S. 315) - über TCP werden verbindungsorientierte Datenübertragungen zwischen den Kommunikationspartnern (Sender und Empfänger) ermöglicht. Das bedeutet, dass während der Datenübertragung permanent spezifische Meldungen ausgetauscht werden.
- **IP⁶** (Heinrich Hübscher, 2011, S. 315) - es ist in der Informationstechnologie weit verbreitetes Netzwerkprotokoll. Im Vergleich mit TCP werden hier verbindungslose Datenübertragungen eingesetzt. Das bedeutet, dass keine dauerhafte Verbindungen zwischen den beiden Kommunikationspartnern besteht. Jeder Rechner im Internet besitzt eine eindeutige IP-Adresse, die einem Host-Namen entspricht.
- **Ethernet-Technik** (IT Wissen Info, 2001) - unter Ethernet versteht man eine Datenübertragungstechnik, die es ermöglicht, Daten innerhalb geschlossener Netzwerke zwischen verschiedenen Geräten zu übertragen. Da sie für lokale Datennetze konzipiert ist, spricht man von LAN-Technik.
- **Netzwerkprotokoll** (Olbrich, 2003, S. 84) - unter Netzwerkprotokoll versteht man ein Kommunikationsprotokoll für den Informationsaustausch zwischen den Teilnehmern in einem Netzwerk. Damit Kommunikationspartner überhaupt miteinander kommunizieren können, müssen bestimmte Vereinbarungen und Regeln eingehalten werden. Diese Regeln sind die Protokolle.

⁵ TCP steht für Transmission Control Protocol

⁶ IP steht für Internet Protocol

- **OSI-Referenzmodell** (Heinrich Hübscher, 2011, S. 274) - das OSI⁷-Referenzmodell besteht aus sieben Schichten und wird daher auch als 7-Schichten-Modell bezeichnet. Jede Schicht besitzt Funktionen und Protokolle, die spezifischen Aufgaben bei der Kommunikation zwischen zwei Systemen erfüllen müssen.
- **Feldbus** (KUNBUS GmbH, 2016) - unter Feldbus versteht man ein leitungsgelinktes Bussystem, das es ermöglicht, Daten in einer Prozessautomatisierungstechnik verschiedener Sensoren und Aktoren zu einem übergeordneten Steuerungsgerät zu übertragen. Eine Vielzahl von Bussystemen ist heute im Einsatz. Die Systeme unterscheiden sich in der Netzarchitektur, Datenübertragungsraten, Datenübertragungsmenge und der Übertragungstechnik.
- **Topologie** (Alexandroff, Paul, Hopf, Heinz, 1945, S. 2) - in Allgemein beschreibt man Topologie als Teilgebiet der Mathematik, welches ursprünglich diejenigen Eigenschaften geometrischer Gebilde behandelt, die bei umkehrbar eindeutigen, stetigen und totalen Abbildungen erhalten bleiben. In der industriellen Kommunikation ist die Topologie laut (Schnabel, 2018) die Verbindungen der teilnehmenden Geräte untereinander, die den Datenverkehr ermöglichen.
- **Controller** (Popp, 2016, S. 381) - ein Controller ist die übergeordnete Steuerung, die typischerweise das Prozessabbild und das Anwenderprogramm enthält. Er ist der aktive Teil bei der Kommunikation der angeschlossenen Devices parametrisiert und konfiguriert wird. Er führt den zyklischen/azyklischen Datenaustausch sowie die Alarmbearbeitung mit den prozessnahen Devices durch.
- **Supervisor** (Popp, 2016, S. 33) - dies kann ein Programmiergerät, Personal Computer oder Human-Machine-Interface-Gerät zu Inbetriebsetzungs- oder Diagnosezwecken sein. Um beispielsweise den Prozess für Testzwecke 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 beispielsweise Diagnose und Alarmer meldet.
- **Knoten & Kanten** - Knoten repräsentieren die Kommunikationspartner und die Kanten die Verbindungen in der industriellen Kommunikation.
- **Logische Topologie** (IT Wissen Info, 2001) - eine logische Topologie beschreibt den Weg, den Daten nehmen, um die Kommunikation in einem Netzwerk zu gewährleisten.
- **Physikalische Topologie** (IT Wissen Info, 2001) - eine physikalische Topologie bezieht sich auf den Aufbau eines Netzes, d.h. in welcher Struktur die einzelnen Kommunikationspartner miteinander verbunden sind. Man kann diese Topologie durchaus mit einer Landkarte vergleichen. Die logischen und physikalischen Topologien müssen nicht identisch sein.

2.2 Grundlegende Netzwerktopologien

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

⁷ OSI steht für Open Systems Interconnection

Vor- und Nachteilen der Topologie kann man auch schon erkennen für welches Einsatzgebiet diese Systeme verwendet werden können.

Topologien werden grafisch mit Knoten und Kanten dargestellt. In großen Netzen findet man oftmals eine Struktur, die sich aus mehreren verschiedenen Topologien zusammensetzt. An dieser Stelle sei noch einmal erwähnt, dass logische Topologie nicht Bestandteil dieser Arbeit sind, sondern nur die physikalische Topologie betrachtet wird.

2.2.1 Linien-Topologie

Die Linien-Topologie war einer der ersten Netzstrukturen, die in kleineren Firmen und privaten Haushalten zu finden war. Ein Koaxialkabel bildet die Basis, die Linie, welche beliebig, bis zu einem gewissen Grad, verlängert werden kann. Die Stationen werden über T-Stücke⁸ an das Kabel angeschlossen. Die angeschlossenen Stationen greifen die Signale vom Kabel ab und senden wo sich das Signal dann in Richtung ausbreitet. Am Anfang und Ende des Kabels sind in der Praxis Widerstände angebracht, um Reflexionen⁹ zu verhindern. Die Linien-Topologie wird auch als Bus-Topologie bezeichnet. Abbildung 3 veranschaulicht diese Topologiemuster mit sieben Stationen.

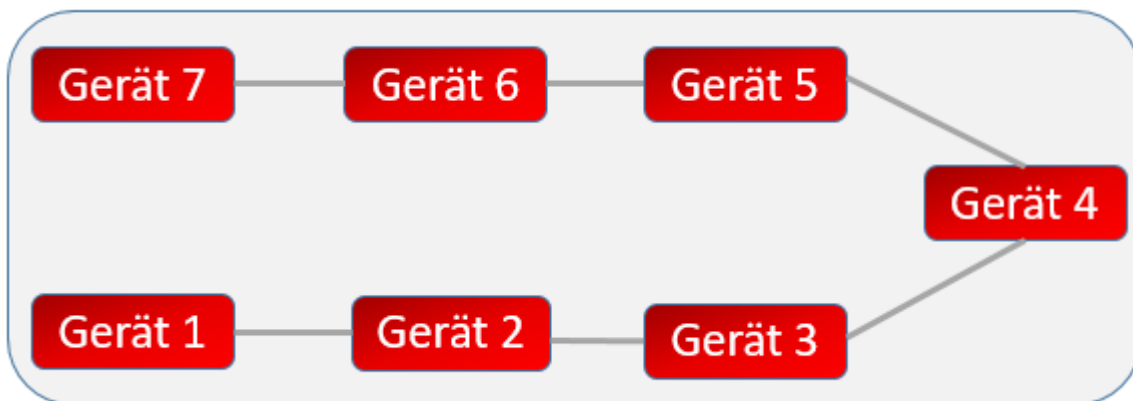


Abbildung 3: Linien-Topologie

Vorteile

- Das System gewährleistet geringe Kosten, da nur geringe Kabelmengen erforderlich sind
- Die einfache Verkabelung über T-Stücke ist sehr sparsam
- Das Netzwerk ist schnell und einfach zu erweitern und dadurch kundenspezifisch präzise anpassbar
- Es werden keine weiteren Geräte zur Übermittlung der Daten benötigt
- Der Ausfall einer einzelnen Station hat für die Funktionalität des übrigen Netzes keine Konsequenzen

Nachteile

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

⁸ T-Stücke sind Verbindungsglieder, die drei Anschlüsse besitzen

⁹ Reflexionen sind ungewollte elektrische Wellen, die auf eine Übertragungsleitung entstehen.

- Es kann immer nur eine Station Daten senden. Während der Sendung sind alle anderen Stationen blockiert, dadurch entsteht den Datenstau
- Aufgrund der Möglichkeit der Datenkollisionen¹⁰ sollte das Medium nur zu ca. 30% ausgelastet werden
- Datenübertragungen können leicht abgehört werden, dadurch entsteht ein Sicherheitsproblem

2.2.2 Ring-Topologie

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

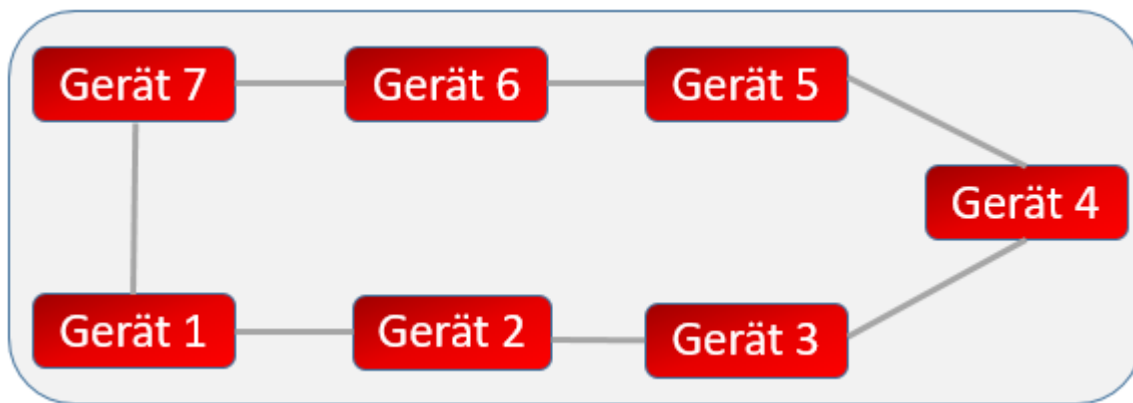


Abbildung 4: Ring-Topologie

Vorteile

- Das System gewährleistet, dass Alle Stationen im gesamten Netzwerkstrang gleiche Zugriffsmöglichkeiten haben
- Da das System eine schnellere Datenübertragung bietet, wird die Übertragungsbandbreite¹¹ garantiert.
- Jede Station arbeitet als Verstärker, d.h. eingehende Daten werden verstärkt und weiterleitet
- Da die Anzahl der Verbindungen pro Station (Grad) bei Erweiterung konstant bleibt, ist das System gut skalierbar
- Das Netz ist leicht programmierbar, da die gesamten Stationen dieselbe Anzahl der Verbindungen pro Station besitzen

Nachteile

- Es entstehen hohe Kosten für den Aufbau des Netzwerks, da die Komponenten teuer sind

¹⁰ Datenkollisionen bezeichnet man die unerwünschten Umstände, dass Daten verloren gehen, wenn zwei oder mehrere Stationen gleichzeitig über das gemeinsame Medium senden

¹¹ Übertragungsbandbreite ist eine Kenngröße, die sich aus in einer Zeiteinheit zu übertragenden Datenmengen ergibt

- Der Durchmesser¹² der Topologie ist relativ hoch
- Höherer Verkabelungsaufwand, da jede Station zwei Verbindungen besitzt

2.2.3 Stern-Topologie

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

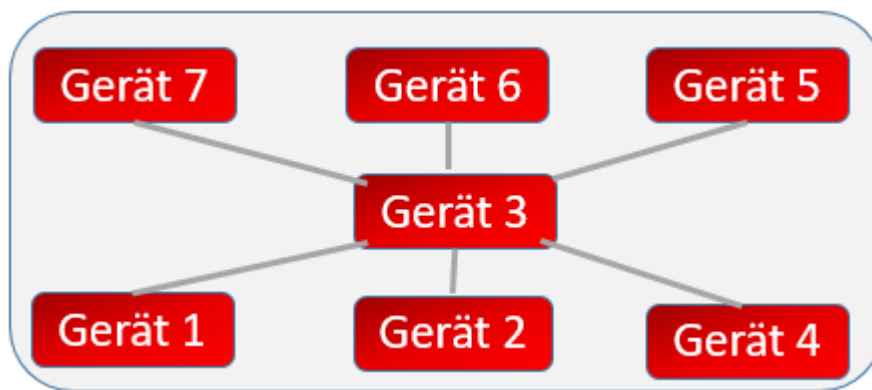


Abbildung 5: Stern-Topologie

Vorteile

- Durch die einfache Verknüpfung der beteiligten Geräte ist das Sternsystem selbsterklärend und leicht verständlich in allen Anwendungsbereichen.
- Das System bietet hohe Übertragungsraten, wenn das zentrale Element ein Switch¹³ ist
- Das System ist leicht erweiterbar und ermöglicht einen sehr flexiblen Betrieb.
- Die Struktur des Systems ermöglicht eine schnelle Fehlersuche hinsichtlich einzelner Geräte und deren Verbindungen.
- Sehr gute Eignung für Broadcast¹⁴-Anwendungen.
- Es gibt kein Routing¹⁵ von Daten, sondern die Daten werden bearbeitet und entsperrenden Geräte weitergeleitet.

Nachteile

- Durch Ausfall des zentralen Elementes wird Netzverkehr unmöglich
- Hohe Kosten entstehen, wenn das zentrale Element leistungsfähig sein muss
- Niedrige Übertragungsrate bei vielen Geräten

¹² Durchmesser versteht man die maximale direkte Entfernung, die zwischen zwei Stationen besteht

¹³ Switch ist ein elektronisches Gerät zur Verbindung mehrerer Stationen in einem lokalen Netzwerk (LAN).

¹⁴ Broadcast ist eine Nachricht, die von einer spezifischen Station zu allen Teilnehmern gleichzeitig übertragen wird.

¹⁵ Routing bezeichnet man als Vorgang, der den Weg zur nächsten Station von Daten bestimmt

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

2.2.4 Baum-Topologie

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

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

Vorteile

- Wenn eine Endstation in Netz ausfällt, ergeben sich daraus keine Konsequenzen für die restlichen Geräte
- Durch die Anordnung der Stationen lässt sich das gesamte Netz strukturell leicht erweitern
- Das Baumsystem gewährleistet eine gute Eignung für spezifische Such- und Sortialgorithmen
- Das System ermöglicht den Transport großer Datenmengen über großen Entfernungen und bietet somit hohes Kosteneinsparpotential

Nachteile

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

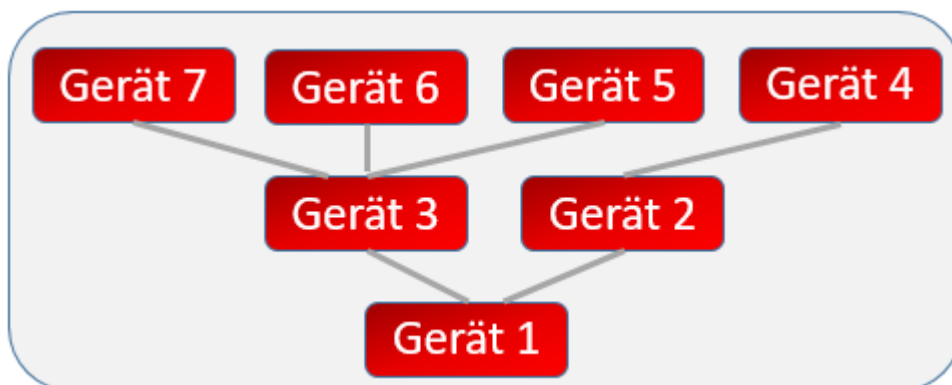


Abbildung 6: Baum-Topologie

2.2.5 Vermaschtes Netz

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



Abbildung 7: Vermaschtes Netzwerk

Vorteile

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

Nachteile

- Dadurch alle Stationen miteinander im Netz verbunden sind, ist der Verdrahtungsaufwand sehr hoch
- Das System erfordert einen sehr hohen Energieverbrauch und verursacht entsprechend steigende Energiekosten
- Für einen zuverlässigen Aufbau benötigt er vergleichsweise komplexes Routing

2.3 Netzwerke-Protokolle

In der industriellen Kommunikation gibt es viele verschiedene Netzwerkprotokolle, die zum Einsatz kommen, beispielsweise PROFIBUS, CANopen, DeviceNet, Sercos, EtherCAT, PROFINET. In diesem Abschnitt werden zwei der wichtigsten Netzwerkprotokolle aus Hilscher Sicht kurz betrachtet. Diese sind PROFIBUS und PROFINET IRT, da auch der Schwerpunkt dieser Arbeit auf diesen beiden Systemen beruht.

2.3.1 PROFIBUS

PROFIBUS war einst ein von SIEMENS (AG, 2018) entwickelter Standard für die Feldbuskommunikation in der Automatisierungstechnik. Es ist heute der universelle Feldbus, der breite

Anwendung in der industriellen Kommunikation, in der Prozessautomatisierung und in der Fabrikautomation, findet. Er ermöglicht die Kopplung von Geräten verschiedener Hersteller ohne besondere Schnittstellenanpassung.

PROFIBUS in das OSI-Modell

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

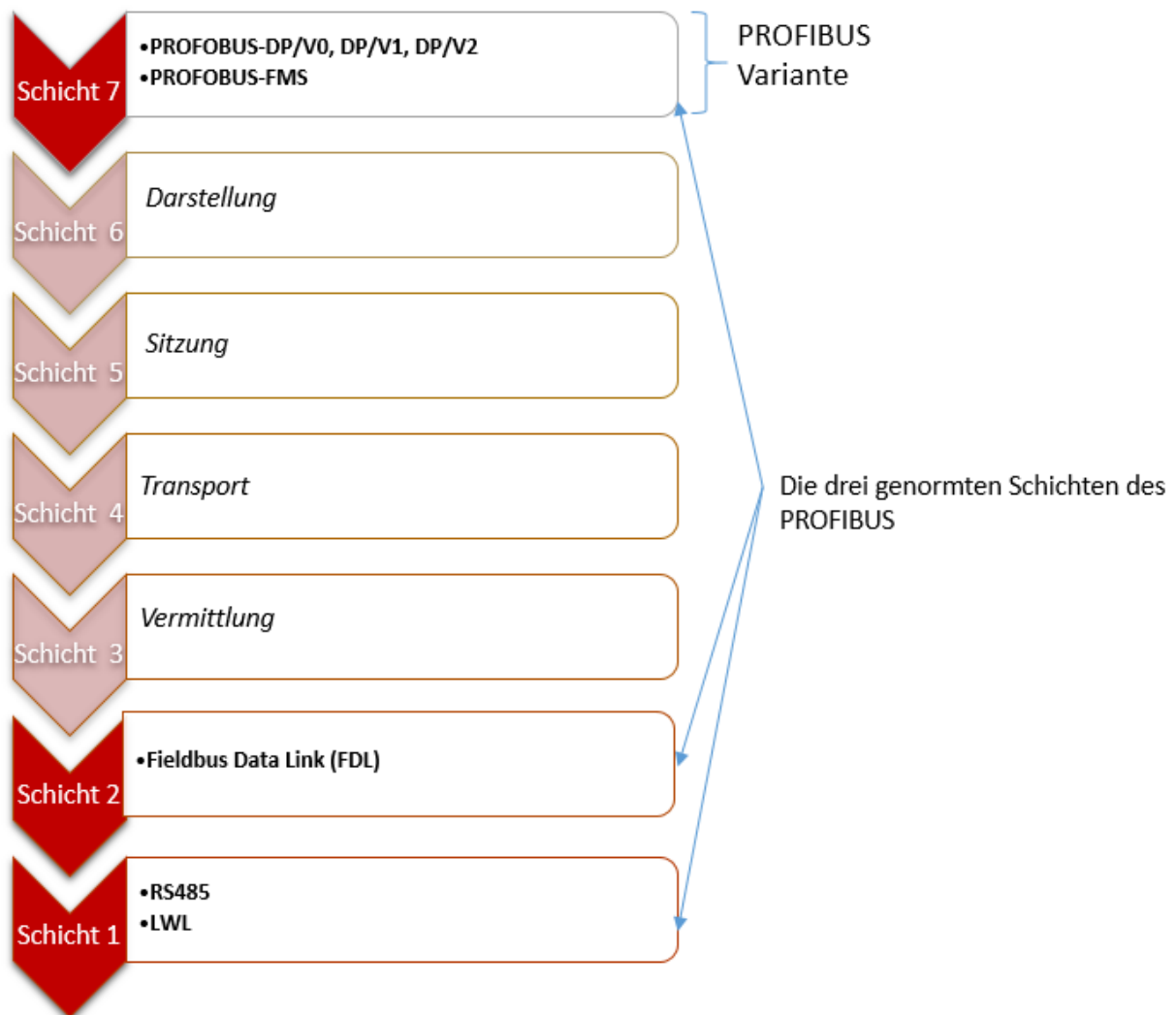


Abbildung 8: PROFIBUS basiert auf international etablierte Standard

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

PROFIBUS Gerätetypen:

- **Master** – er ist typischerweise eine Steuerung, die den gesamten Datenverkehr initiiert. Er greift auf Resources zu und versendet Nachrichten ohne externe Aufforderungen.

- **Slave** – er ist in einem Feldbussystem ein passiver Kommunikationsteilnehmer, der von dem Master gesteuert wird. Er muss für einen Buszugriff direkt von dem Master aufgefordert werden.

PROFIBUS Varianten:

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

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

Mit dem PROFIBUS können folgende Systeme realisiert werden:

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

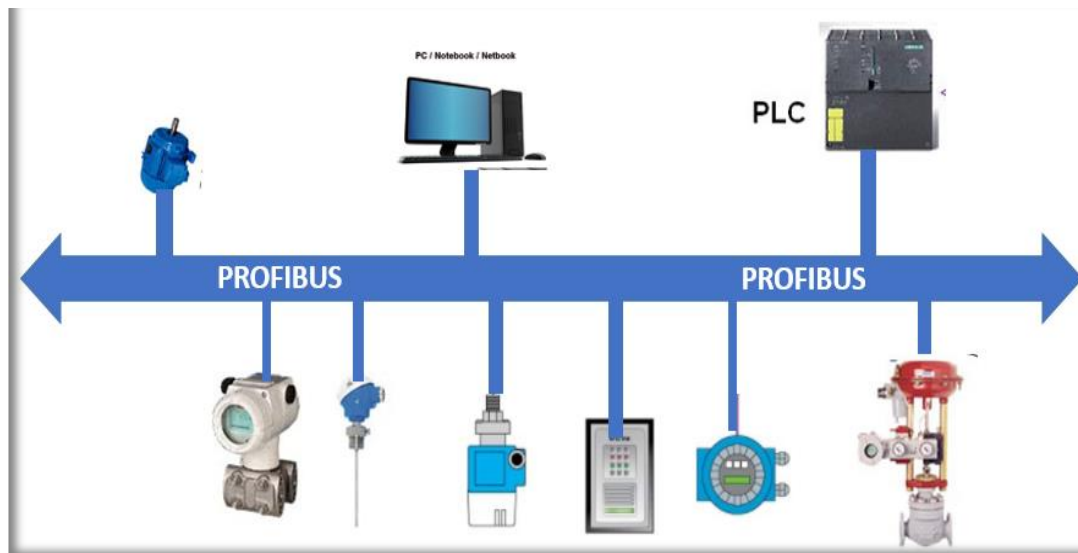


Abbildung 9: Beispielhafte PROFIBUS-Topologie an einem Master-Slave-System

Abbildung 9 zeigt ein PROFIBUS PA-Topologie auf Basis eines Master-Slave-Systems. Das System besteht aus einer Master-Station (SPS/PLC) und den angeschlossenen Slave-Stationen. In diesem System können zyklischen als auch azyklische Daten zwischen dem Master und den Slaves ausgetauscht werden. Die SPS wird zur Steuerung des gesamten Systems und der PC zur Konfiguration des Systems und zur Programmierung der SPS verwendet.

2.3.2 PROFINET

PROFINET wurde in Zusammenarbeit von Siemens und den Mitgliedsfirmen der Profibus-Nutzerorganisation (PNO) entwickelt und wird von PROFIBUS und PROFINET International (PI) gewartet und weiterentwickelt. Basierend auf Ethernet 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 Fabrikautomation eingesetzt.

PROFINET Gerätetypen:

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

PROFINET Varianten:

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

Auch beim PROFINET werden die Funktionalitäten und Eigenschaften eines PROFINET Device in einer Gerätebeschreibungsdatei beschrieben. Diese Datei enthält neben den relevanten Daten für das Engineering¹⁶ auch Daten, die für den Datenaustausch mit dem Device von Bedeutung sind. Die Nutzerorganisation schreibt zwingend die Lieferung einer GSDML-Datei vor. Diese ist Bestandteil des Zertifizierungstests.

¹⁶ widmet sich der technischen Forschung und Entwicklung, sowie der Produktionstechnik in der industriellen Kommunikation auf naturwissenschaftlicher Grundlage

3 Ausgangssituation

Die Firma Hilscher hat die Fachkompetenz für die Entwicklung und Produktion industrieller Kommunikationslösungen für die moderne Fabrikautomation. Die Produkte aus dem Hilscher-Portfolio unterstützen alle in diesem Umfeld etablierten Netzwerkprotokolle. Als ein Systemanbieter hat die Firma Hilscher für die Konfiguration und Inbetriebnahme der Netzwerke und Gerätegruppen entsprechende Softwarelösungen mit im Angebot. Diese werden aktuell auf eine neue technologische Basis gestellt. Die neuentwickelte Software heißt Communication Studio, die in Zukunft unter anderem auch eine web-basierte Konfiguration ermöglichen soll. Durch den modularen Aufbau der Software ist es möglich, den Topology-Editor als Web-Topology-Editor zu konzipieren.

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

3.1 FDT

FDT (René Simon, 2003) ist eine Technologie, die für einen durchgängigen und einheitlichen Datenaustausch zwischen Feldgeräten und Automatisierungssystemen konzipiert ist. Sie ermöglicht den Zugriff auf die Feldgeräte-Parameter und –Funktionen von einem zentralen Raum/Rechner aus, aus einer einzigen Applikation heraus. Die FDT ist reine Software-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. Durch die Standardisierung der FDT-Schnittstellen 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 entsprechende Annexe z. B. für PROFINET und PROFIBUS (siehe Abschnitt 3.4), Ethernet IP, DeviceNET, HART und CANopen oder durch herstellerspezifische Protokollintegrationen bereitgestellt. Die aktuelle Version ist FDT2.0.

3.2 Was ist Communication Studio?

Communication Studio, in dieser Arbeit abgekürzt auch ComStudio genannt, ist der Name für das neue Softwareprodukt der Firma Hilscher. Diese Software soll zukünftig die bisherige Software SYCON.net ablösen. Grund für die Ablösung der Software durch das ComStudio sind unter anderem die schlechte Erweiterbarkeit und dadurch in Zukunft fehlende Funktionalität. Das ComStudio basiert vollständig auf FDT2.0-Technologie, ist modular aufgebaut und besitzt die Möglichkeit, Softwaremodule über eine Plugin-Schnittstelle zu integrieren. Vorrangig wurde für die Programmierung C#¹⁷ verwendet und die Oberflächen wurden mit WPF¹⁸ erstellt. Durch geringfügige Anpassungen konnten einige bestehenden Konfigurationsmodule aus der Software SYCON.net im ComStudio wiederverwendet werden. Andere Teile, wie z.B. die Diagnose der Baugruppen wurden neu entwickelt. Dieser Programmteil wurde als „Web-Diagnostic“ als Client für browserbasierte Anwendungen auf HTML-, Angular- oder TypeScript-Basis erstellt. Ein Grund hierfür ist die häufigere Nutzung ohne Änderung der Konfiguration und somit ist ein Start der gesamten Software nicht nötig.

Abbildung 10 zeigt die wichtigsten Module des „Communication Studios“. Diese Module bieten den Vorteil, dass die Software mit den steigenden Anforderungen mitwachsen kann.

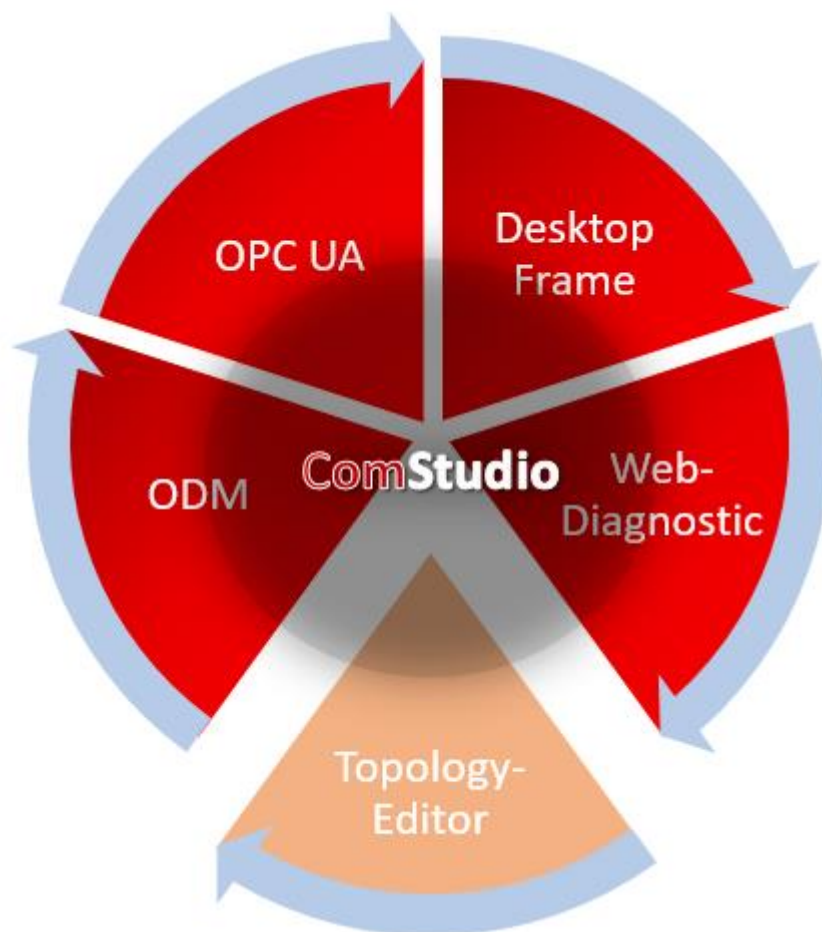


Abbildung 10: Vereinfachtes Übersichtsdiagramm ComStudio mit Topology-Editor

¹⁷ C# ist eine objektorientierte Programmiersprache von der Firma Microsoft und ist plattformunabhängig.

¹⁸ WPF ist eine Klassenbibliothek zur Entwicklung von grafischen Benutzeroberflächen.

3.3 Befragungsmethoden

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

- **Regelmäßige Meetings**
Bei dieser Methode werden die direkten mündlichen Fragen gestellt. Die Befragung wird entweder auf Basis eines Fragenkatalogs durchgeführt oder Fragen, die sich im Verlauf des Meetings ergeben.
- **Dokumentenanalyse**
Bei dieser Analyse werden Informationen aus Spezifikationen ermittelt und die Anforderungen der bisherigeren Softwareanwendung ComStudio extrahiert.
- **Beobachtungen**
Durch Beobachtung werden die Sachverhalte durch sinnliche Wahrnehmung aufgenommen. Ich habe den Mitarbeitern bei der Arbeit und beim Umgang mit dem ComStudio zugesehen. Aus diesen Beobachtungen wurden viele Information gewonnen wie z.B. die Reihenfolge der Bedienung / Fenster

3.4 Auswertung

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

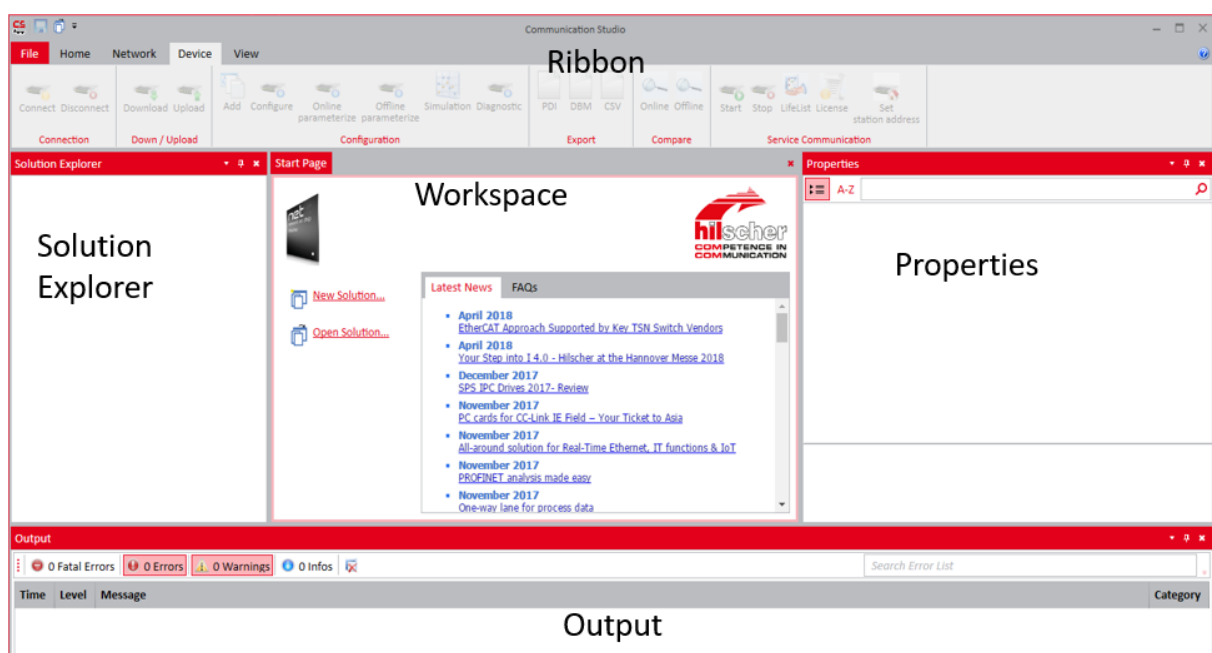


Abbildung 11: Benutzeroberfläche von ComStudio mit spezifischen Ansichten

Abbildung 11 zeigt die Standard-Ansichten der bestehenden Software, hier sieht der Nutzer die Liste der Projekte, welche in einer Solution zusammengefasst sind, detaillierte Information von spezifischen Gerätetypen bzw. Verbindungstypen, Ergebnisse in bereits geschlossenen Prozessen und kann in Workspace eine neue GUI öffnen.

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

- Das bisherige System bietet nur eine hierarchische Darstellung der Geräte im sogenannten „*Solution Explorer*“ (Abbildung 11) an. Allerdings ist diese Art der Darstellung kein Ersatz für eine topologische Darstellung eines Netzwerkes
- Die Darstellung, Bearbeitung und das Handling der Topologie soll einheitlich für alle Netzwerke sein
- ComStudio ist nicht cloud-fähig. Die Firma Hilscher sieht zukünftig die Möglichkeit des Einsatzes von Cloud-Konfiguratoren
- Es wird ein modularer Aufbau mit Plugin-Schnittstelle gewährleistet
- Topologie wird zur Konfigurations- und Diagnosezeit benötigt.
- Das System bietet schon eine Möglichkeit, die detaillierte Information über die Kommunikationsgeräte, die Kommunikationsinfrastrukturgeräte und Verbindungen in einem sogenannten „*Properties*“-Fenster. (Abbildung 11)
- Das System bietet eine Multifunktionsleiste („*Ribbon*“), die Elemente Menüsteuerung, Symbolleisten und Dialoge miteinander verbindet, sie widerspiegelt 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 für eine realistische Kosteneinschätzung in Projekten (Sommerville, 2007).

Die folgenden Punkte listen einige Vorteile, warum wir überhaupt diese Analyse grundsätzlich durchgeführt haben:

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

Dieser Abschnitt beschreibt 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 (FR¹⁹) spezifizieren die Funktionalität oder das Verhalten des Softwareproduktes unter festgelegten Bedingungen oder als Frage „Was soll das Softwareprodukt tun?“.

Hierbei gibt es die Muss-Kriterien, das heißt die Kriterien, die auf jeden Fall umgesetzt werden müssen. Sie beschreiben vor allem den Einsatz essentieller Funktionen des Topology-Editors. Weiterhin formuliert man die Kann-Kriterien, die umgesetzt werden, wenn es die Ressourcen und die Planung des Projektes zulassen und nicht die Einsatzfähigkeiten des Systems beeinträchtigen.

4.1.1 Muss-Kriterien

FR010: Die Topologie ist in einer Ansicht darzustellen.

FR020: Die Topologie-Darstellung kann separat geschlossen werden.

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

¹⁹ Functional Requirement

| | |
|--------|---|
| 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 der GUI-Sicht 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 | Unterstützung für Zoom-Funtionen. |
| FR120 | Persistenz der Daten (Speichern, Laden) |

4.1.2 Kann-Kriterien

| | |
|--------|--|
| FR200: | Bereitstellen einer „Über Uns“-Ansicht. |
| FR210: | Funktion zum Umstellen der Bediensprache, Lokalisierung. |
| FR220: | Funktion zur Umschaltung der Topologiemuster im zulässigen Kontext (siehe Abschnitt 2.2) |
| FR230 | Funktion zum Suchen und Ersetzen von Informationen. |
| FR240 | Funktion zur Filterung von Informationen und der Darstellung. |
| FR250 | Bereitstellen der Topologieinformationen 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 (NFRs²⁰), 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 Softwarekomponente ist ein Bestandteil des Communication Studio.
- NF070: Der Datenaustausch erfolgt im XML-Format.
- NF080: Die Softwarekomponente 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 häufig genutzten Funktionsumfang geben zu können, soll nachfolgend eine Auswahl von Anwendungsfällen dargestellt und erörtert werden. Abbildung 12 gibt bereits einen Überblick über einige der Anforderungen.

Prinzipiell lassen sich bei der Interaktion mit dem System zwei Nutzerrollen identifizieren. Bei Anzeigen und Erstellen der Netzwerktopologie sind hier der Kunde und der Inbetriebnehmer zu nennen.

²⁰ Nonfunctional Requirement

Der Endanwender der Software möchte im System eine Topologie anzeigen. Er soll im Communication Studio das entsprechende Projekt anlegen oder ein bestehendes Projekt importieren. Nach erfolgreichem Einfügen der DTMs, kann der Anwender die Topologie anzeigen lassen. Somit kann er hier die entsprechende Netzwerktopologie bearbeiten.

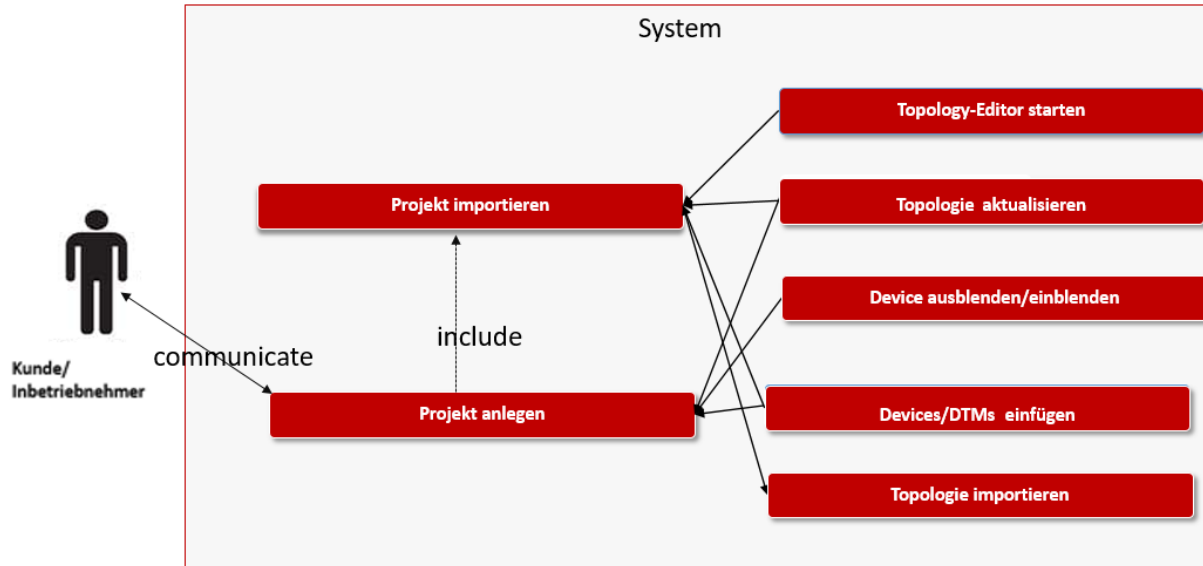


Abbildung 12: Anwendungsfall-Diagramm: Ausgewählte Funktionen des Systems

Während der Bearbeitung haben die Anwender prinzipiell zwei Optionen:

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

5 Evaluation der Technologien

Topology-Editor basiert auf einem verteilten System, das Module zur Umgebungserfassung mit Datenverarbeitungsknoten und Schnittstellen zu andere Programmen kombiniert. Der Entwicklungsprozess der Software für ein solches System umfasst neben der eigentlichen Anwendungsentwicklung zwei periphere Aufgaben: Zum einen muss die Kommunikation zwischen den Komponenten des Systems sichergestellt und zum anderen die Interfaces zur Hardware oder anderen externen Programmen bereitgestellt werden.

Um den Entwicklungsaufwand an dieser Stelle zu reduzieren, muss die in Kapitel 4 beschriebene Vielfalt gekapselt und der Zugriff auf eine uniforme Schnittstelle abgebildet werden. Erst damit ist eine Austauschbarkeit und Wiederverwendbarkeit von Komponenten umsetzbar. Dadurch kann eines der wichtigsten Muss-Kriterien aus dem Anforderungskatalog abgedeckt werden.

Es wurde eine Fülle von verschiedenen Frameworks für die Entwicklung des Plugins ausfindig gemacht. Nach dem Abstecken des Szenarienkontexts, wie zum Beispiel der beteiligten Komponenten, den nötigen Algorithmen, den erforderlichen Kommunikationseigenschaften usw., bedurfte es aufwändiger Recherche, um aus der Vielzahl der Verfügbaren Optionen die am besten geeignete zu wählen. Dieses Kapitel dient insbesondere für die clientseitige²¹ Umsetzung mit Vergleichsaufstellung als Überblick über die infrage kommenden Frameworks.

Für die Visualisierung von Informationen werden die Bibliotheken GoJS (Northwoods Software, 1998), KendoUI (Progress Software Corporation , 2018) und SAPUI5 (SAP, 2018) evaluiert.

Für das Styling und die Strukturierung der Benutzeroberflächen soll das Bootstrap Frontend-CSS-Framework (Twitter, 2010) verwendet werden. Für die serverseitige²² Umsetzung soll die ASP.NET.Core 2.0 verwendet werden. Diese werden bereits für andere Komponenten der Ziel-Host Applikation ComStudio eingesetzt.

5.1 Clientseitig 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 dieser Komponenten in einem Gesamtsystem zu ermöglichen und dabei den Software-Entwicklungsprozess zu vereinfachen und zu beschleunigen, sind viele

²¹ beschreibt ein Computerprogramm, das Dienste von einem Server abrufen

²² beschreibt ein Computerprogramm, welches auf einem zentralen Computer ausgeführt wird

Frameworks mit unterschiedlichen Schwerpunkten entwickelt worden. Diese gehen zum Beispiel Kommunikation und Konfiguration, die mit einem modularen System einhergehen, an.

Das Entwickeln und Erstellen von HTML-Seiten²³, die flexibel und einfach zu warten sind, kann eine Herausforderung darstellen. 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 konzipieren, dass sie flexibel ist und im Laufe der Zeit leicht geändert oder erweitert werden kann. Entwerfen für diese Art von Flexibilität kann schwer zu erreichen sein, 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 beeinträchtigen.

Im Folgenden stützt sich dieser Abschnitt auf die wichtigsten Frameworks, die zum einen eine Erfüllung der in Abschnitt 4.1 und 4.2 beschriebenen Aspekte bieten und zum anderen eine Verbreitung gefunden haben.

5.1.1 JavaScript Frameworks

Angular

Angular ist eine Entwicklungsplattform für die Erstellung von mobilen und Desktop-Webanwendungen mit TypeScript / JavaScript und anderen Sprachen. Es wurde durch Google angeführt und ist heute von einer Community aus Einzelpersonen und Unternehmen entwickelt. Seine Architektur besteht aus vielen unabhängigen Bausteinen. Im folgenden wird eine kurze Übersicht über den wichtigsten Bausteinen gesprochen:

- **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 mehreren Modulen. Sie können verwendet werden, um bestimmte geschäftsbezogenen Funktionen (z.B. Editieren von Topologie, Startseite verwalten) darzustellen und alle Ansichten, Dienste und Datenmodelle zu kapseln, die zum Implementieren dieser Funktionalität erforderlich sind.
- **Component:** Bei Angular ist keine direkte Manipulation des HTML-DOM (MDN web docs, 2005) vorgesehen, sondern eine Trennung nach MVC-Muster²⁴. Der Entwickler definiert dafür ein Tag²⁵ in einer Seite, der durch eine mit *@Component* dekorierte Komponentenklasse repräsentiert. Der *Selector* legt den Tagnamen fest. Die mit der Komponente verbundene Vorlage (*Template*) kann der Webentwickler direkt als Zeichenkette in der Eigenschaft *template* angeben, was sich aber nur bei sehr kurzen HTML-Blöcken anbietet. Eine Komponente kann auch eigene CSS-Vorlagen (w3schools.com, 1999) (Inline oder als eigenständige Datei) besitzen, die allein für die Darstellung dieser Komponente gelten. Ein wichtiges Verhalten der Komponenten ist, dass sie 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

²³ sind Seiten zur Strukturierung digitaler Dokumente mit Inhalten wie Hyperlinks, Bildern usw.

²⁴ ist ein Architekturmuster zur Trennung von Software in drei Komponenten (Datenmodell, Ansicht und Programmsteuerung)

²⁵ ist ein HTML-Anweisung in spitzen Klammern

einer Ansicht bietet den Vorteil, dass man nicht jedes Detail der Ansicht erstellen muss. Angular Templates erlauben unterschiedliche Syntaxen wie Interpolation, Template Statement usw. Als eigenständige Datei lässt sich die Angular Template in einer Applikation durch die Dateienden *.html* erkennen.

Vue.js

Vue ist ein fortschrittliches Framework für den Aufbau von Benutzeroberflächen, das von Evan You (LinkedIn, 2018) im Jahr 2013 entwickelt wurde. Die Aktuelle Version ist 2.5.16 (Vue.js, 2018). Es ist von Grund auf so konzipiert, dass es stufenweise adaptierbar 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 SPA²⁶-Anwendungen unterstützt. Dieses Framework arbeitet nach dem Entwurfsmuster MVVM und ist stark von Angular inspiriert.

React

React ist ein JavaScript-Framework zur Erstellung von Webanwendungen. Es wird seit 2013 von Facebook unter einer Open Source-Lizenz entwickelt und besitzt eine große Online Community mit vielen helfenden Entwicklern und Unterstützern. Die aktuelle Version 16.3.2 (Facebook/React, 2018) besitzt eine deutliche Verbesserung der Fehlermeldung. Es handelt sich um ein clientseitiges Frontend-Framework, das heißt, der Quelltext wird für die Ausführung im Browser geschrieben und dient der Implementierung einer Benutzeroberfläche. Das Hauptprinzip ist dabei die Erweiterung des Vokabulars von HTML, damit dieses um Funktionalitäten erweitert wird, die es ermöglichen, dynamische Webanwendungen zu erstellen.

In dynamischen Webseiten und -anwendungen können beispielsweise Quelltext-Ausschnitte in verschiedene Seiten eingebunden werden, um Redundanzen zu vermeiden. Außerdem ermöglicht eine dynamische Seite die Interaktion des Nutzers mit bestimmten Inhalten. Das Gegenteil sind statische Webseiten, die aus unveränderlichen und zusammenhangslosen HTML-Dokumenten bestehen. Die einzige Möglichkeit der Navigation bildet dort die Nutzung von Hyperlinks.

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

5.1.2 UI-JavaScript-Bibliotheken

Für die Entwicklung der grafischen Benutzerschnittstelle kommt eine Vielzahl an unterschiedlicher externen Bibliotheken zum Einsatz. Darunter sind Klassen für Controls, Dokumentation, Layout-Panels, Benutzerführung Animationen oder Klassen zum Einbinden von Diagramme und Bildern. Oft werden diese externen Bibliotheken als User Interface-Bibliotheken bezeichnet.

Um die geeigneteren User Interface-Bibliotheken für des Plugins zu finden, werden in den folgenden Abschnitten einigen externen Bibliotheken zur Visualisierung von Informationen angegangen.

²⁶ steht für Single-Page Application

SAPUI5

SAPUI5 ist ein JavaScript-Framework, das von der Firma SAP (SAP, 2018) entwickelt wurde. Es besteht aus einem funktionsreichen Kern und einer sehr großen Anzahl von UI-Steuererelementen, die in einer Handvoll Bibliotheken organisiert sind.

Diese Bibliotheken besitzen vor allem eine effiziente Engine zum Erstellen und Aktualisieren der HTML-Steuererelemente, darüber hinaus unterstützen sie das MVC-Konzept und die deklarative UI²⁷-Konstruktion.

Die Steuererelemente reichen von einfachen Button-Steuererelementen bis zu komplexen Steuererelementen, beispielsweise Grid-System und Diagram-Steuererelementen. Für Entwickler bietet SAPUI5 leistungsstarke Konzepte:

- eine solide Grundlage, die die Entwicklungsprozesse vereinfacht, indem viele Aspekte der modernen Entwicklung hinter den Kulissen verwaltet werden.
- integrierte Unterstützung für Architekturkonzepte wie Zwei-Wege-Datenbindung und Routing
- die Bindung mit JSON, XML und anderen Datenformaten

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

Kendo UI

Das bulgarische Consulting-Unternehmen Telerik (Progress Software Corporation, 2018) bietet die Kendo UI-Framework für mobile und webbasierte Einsatzfelder an. Der Code läuft nicht in einer WebView²⁸, sondern in einer Runtime, die von Telerik-Ingenieuren zur Interaktion mit dem zugrundeliegenden Betriebssystemen entwickelt wurde.

Daher können Sie auf die Steuererelemente des UI-Stacks der jeweils Plattform zurückgreifen. Kendo UI bietet zudem – zur Reduktion des Portierungsaufwands – die Möglichkeit, die Bedienschnittstelle in einer XML angelegten Sprache zu beschreiben. Die Laufzeitumgebung erzeugt daraus eine Widget²⁹-Struktur.

Eine gut gemachte Kendo-UI Applikation unterscheidet sich daher nicht wesentlich von einer komplett nativen und lässt sich zudem einfach zwischen Betriebssystemen portieren.

Vorteile

Kendo UI

- liefert ein umfangreiches Framework für die responsive Webentwicklung.
- bietet für viele Komponenten eine gute Basis, denn browserkompatibilität und responsives Verhalten sind feste Bestandteile des Frameworks
- enthält neben den Standard-Komponenten unterschiedliche Farbschemata
- bringt eine beeindruckende, ausführliche Dokumentation aller Komponenten mit ihren Optionen mit

Nachteile

- Das Laden der Kendo UI-Bibliothek dauert länger

²⁷ Steht für User Interface

²⁸ ist eine Komponente zur Darstellung von Web-Inhalten, z.B. HTML-Dateien mit JavaScript und CSS

²⁹ ist eine Komponente eines grafischen Fenstersystems.

- Kendo UI generiert hauptsächlich Markup über JavaScript, was die Anpassung der Klasse erschwert
- Kendo UI ist nicht geeignet für mobile Geräten

GoJS

GoJS (Northwoods Software, 1998) ist eine JavaScript-Bibliothek, mit der auf einfache Weise interaktive Diagramme in modernen Webbrowsern erstellt werden können. Sie unterstützt grafische Vorlagen und Datenbindung von grafischen Objekteigenschaften zum Modellieren von Daten.

Die Anpassung von Aussehen und Verhalten hängt hauptsächlich von der Einstellung der Eigenschaften ab.

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

Vorteile

Die wesentlichen Vorteile liegen bei dem Einsatz von GoJS-Diagramm in der breiten Unterstützung der Darstellung von Netzwerkstruktur in Webbrowsern.

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. Softwareentwicklern wird ein Freiraum eingeräumt, nach eigenen Vorstellungen zu modellieren. Besonders Vorteilhaft ist die Tatsache, dass GoJS auch die Erweiterungsmöglichkeiten durch TypeScript vorsieht und damit wirklich jede Möglichkeit bietet, den TypeScript-Notationsrahmen zu erschaffen. Diese Möglichkeit sollte aber nur in äußersten Fällen (wenn z.B. JavaScript nicht 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. Sie ist zudem Open-Source ist.

Nachteile

Die Nachteile lassen sich nun wiederum aus dem Umfang der Bibliothek und den Preisen ableiten. GoJS ist sehr vielfältig, so dass die lernkurve steil ist. Außerdem wird man in der Regel feststellen, dass viele Elemente sehr selten zum Einsatz kommen und damit eher als Ballast der Bibliothek angesehen werden. Um eine Lizenz zu bekommen, soll man pro Entwickler über 2500 Euro (Northwoods Software, 1998) ausgeben, was zu teuer ist.

5.2 Vergleichskriterien

Nachdem in vorangegangenen 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. Abschließend werden die Programmierkonzepte betrachtet.

5.2.1 Lernkurve und Kosten

In der industriellen Kommunikation integrieren Konfigurationsapplikationen vielfältige Feldbussysteme mit unterschiedlichen Eigenschaften. Konfiguratoren für industrielle Kommunikationslösungen müssen heterogene Netzwerke mit vielfältigen Protokollen unterstützen. Um dennoch Applikationen auf Support-Aufkommen sowohl für den Endanwender als auch für die Entwickler reduzieren zu können, spielen den Erfolgsgrad des Lernens und Kosten eines Frameworks eine wichtige Rolle.

- **Lernkurve:** stellt die Entwicklung eines Trainierenden beim Erlernen eines Frameworks dar, spiegelt somit den Erfolgsgrad des Lernens über den Verlauf der Zeit. Da die Masterarbeit in einer zeitlichen Periode begrenzt ist, sollte diese leicht und schnell wie möglich sein.
- **Kosten:** hier sind alle finanzielle Ausgaben gemeint, die für die erfolgswirksame Nutzung von einem Framework entstehen. 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 der 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 Nachrichtenaustausch zwischen Komponenten erfolgt und in welcher Beziehung die Komponenten dabei zueinanderstehen. Die Interaktion kann z.B. über eine Event-Kommunikation oder 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 gewährleistet werden kann, 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 andererseits die Anwendungsentwicklung eben mit dieser betreffen.

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

die Anwendbarkeit der Software definiert. Bestimmte Lizenzmodelle erleichtern die Entwicklungsarbeit und bieten in Wikis eine Vielzahl von Antworten, Anregungen und Beispielcodes an.

5.3 Ergebnisse der Analyse

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

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

Tabelle 2: Vergleichstabelle für JavaScript-Framework

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

Tabelle 3: Vergleichstabelle für UI-Bibliotheken

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

In der Praxis stellen die Frameworks eine Steigerung der Produktivität dar, ermöglichen schnelle Produktentwicklung und deren Markteinführung.

5.4 Bootstrap

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

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

Die Idee hinter dem Framework ist, die Designer und die Entwickler auf einen gemeinsamen Nenner zu bringen.

Bootstrap wurde mit Augenmerk auf HTML5 und CSS3 entwickelt, um so von den aktuellsten Funktionsweisen profitieren zu können, deutlich wird dies beispielsweise bei den in CSS3 eingeführten Eigenschaften für abgerundete Ecken, Farbverläufen und Schatten.

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

Vorteile:

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

Nachteile:

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

5.5 ASP.NET Core

Bei Topology-Editor wird die Interaktion zwischen Client und Server durch den Austausch JSON-basierter Nachrichten geschehen, die mittels http-Protokoll übertragen werden. Da Web Services ein Internetdienst ist, müssen die eingesetzten Technologien Plattformunabhängig und unabhängig von einer bestimmten Programmiersprache sein. REST³⁰ ist eine Möglichkeit, um Web Services zu implementieren. Da wir die C#-Programmiersprache gewählt haben, werden wir als Framework ASP.NET Core 2.0 (Microsoft Corporation, 2018) von Microsoft nehmen.

REST

REST ist plattform-und programmiersprachenunabhängig, und ist einer der verbreitetsten Möglichkeiten, um Web Services zu realisieren, er ist das Architekturvorbild für das Internet und ist geeignet für die Erstellung von Web Services, und ist kein Standard wie SOAP³¹.

Ein REST-System besteht aus Resources, die per URI³² 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

³⁰ Representational State Transfer

³¹ Simple Object Access Protocol: Protokoll zum Austausch strukturierter Informationen

³² Uniform Resource Identifier

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

ASP.NET Core 2.0

ASP.NET Core 2.0 ist ein plattformübergreifendes, leistungsstarkes Open-Source-Framework zum Erstellen moderner, cloudbasierter, mit dem Internet verbundener Anwendungen. Bei ASP.NET Core 2.0 handelt es sich um eine Neugestaltung des Frameworks 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
- Ist Open Source und mit Fokus auf der Community
- ASP.NET Core besteht vollständig aus NuGet-Paketen.

6 Spezifikation und Konzept

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

6.1 Topology-Editor, grundlegende Konzepte

Topology-Editor ist keine Anforderung an FDT-Frame-Applikation. Es bietet eine Benutzerfreundlichkeit für die Visualisierung einer Netzwerktopologie. In diesem Abschnitt werden die grundlegenden Konzepte und Implementierungsansätze für das Plugin. Eine Rahmenanwendung wie ComStudio, die das Topology-Editor-Plugin hosten soll, soll als Datenquelle fungieren.

Topology-Editor-Plugin besteht aus zwei Hauptkomponenten: der Topology-Editor-Server und Topology-Editor-Client. Der Topology-Editor-Server umfasst Kernfunktionen wie das Parsen der protokollneutralen Gerätebeschreibungen, das Erstellen, Aktualisieren und Löschen eines Topologie-Modells für die geparschten Informationen. Darüber hinaus bietet er einen Mechanismus für Fehlerbehandlungen.

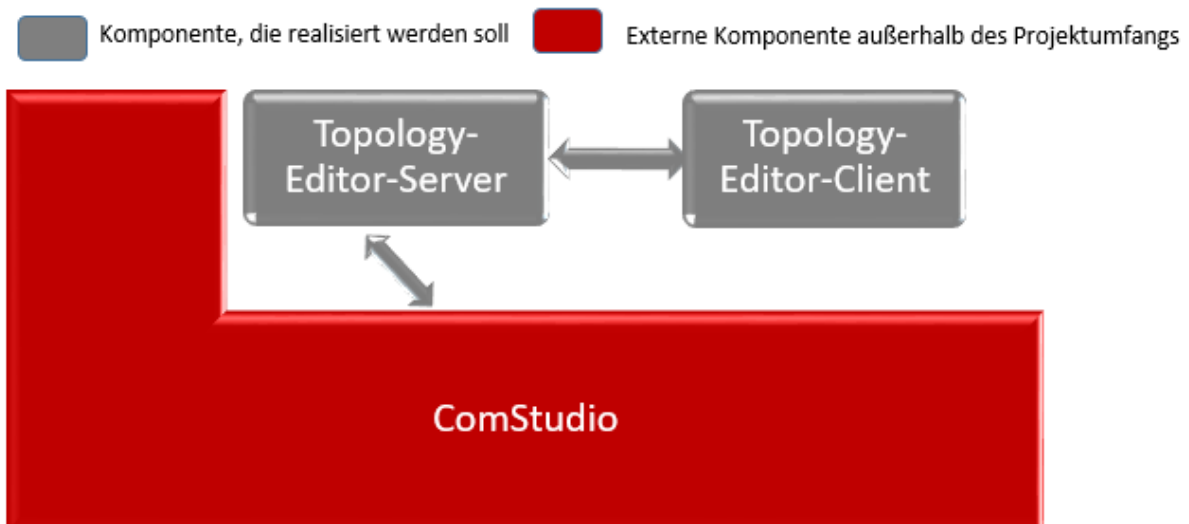


Abbildung 13: ComStudio mit dem Topology-Editor Komponenten

Topology-Editor-Client ist die clientseitige Anwendung, welche das HTTP-Protokoll zur Datenkommunikation mit dem Server verwenden, um Topologie-Informationen abzufragen und sie anschließend für den Nutzer grafisch darzustellen. Der Client kann direkt in ComStudio gehostet werden oder als Web-Anwendungen verwendet werden. Abbildung 13 zeigt das Zusammenspiel zwischen ComStudio und Topology-Editor und stellt die zu realisierenden Komponenten dar.

Die Komponente für die Benutzeroberflächen hängt von der Geschäftslogik des Servers ab und verwendet ihre Funktionen. Im Gegensatz dazu sollte die Geschäftslogik nicht von der Benutzerschnittstellenkomponente abhängen.

Die Bestandteile des Topology-Editors müssen in verschiedenen Szenarien Feldbus-neutral verwendbar sein. Ein dediziertes Szenario ist das ComStudio-Szenario. Dieses soll von der Lösung unterstützt werden.

6.2 ComStudio-Szenario

Topology-Editor-Server und Topology-Editor-Client sind in der Host-Applikation ComStudio eingebettet. Die feldbusspezifischen Geräte werden in ComStudio nach dem FDT-Standard verwendet. Dieser Ansatz ist sinnvoll für Szenarien, in denen ein Benutzer anspruchsvollere Topologien für seine feldbusspezifischen Geräte bereitstellen möchte ohne Fachwissen über die jeweils verwendeten Protokolle beherrschen zu müssen oder wenn Funktionen zum Darstellen einer physikalischen Anordnung (der zulässigen Verknüpfungen) der protokollabhängigen Geräte mit Datenelementen sowie zum Lesen, Importieren, Exportieren, Löschen, Schreiben und Überwachen dieser Elemente für Datenänderungen.

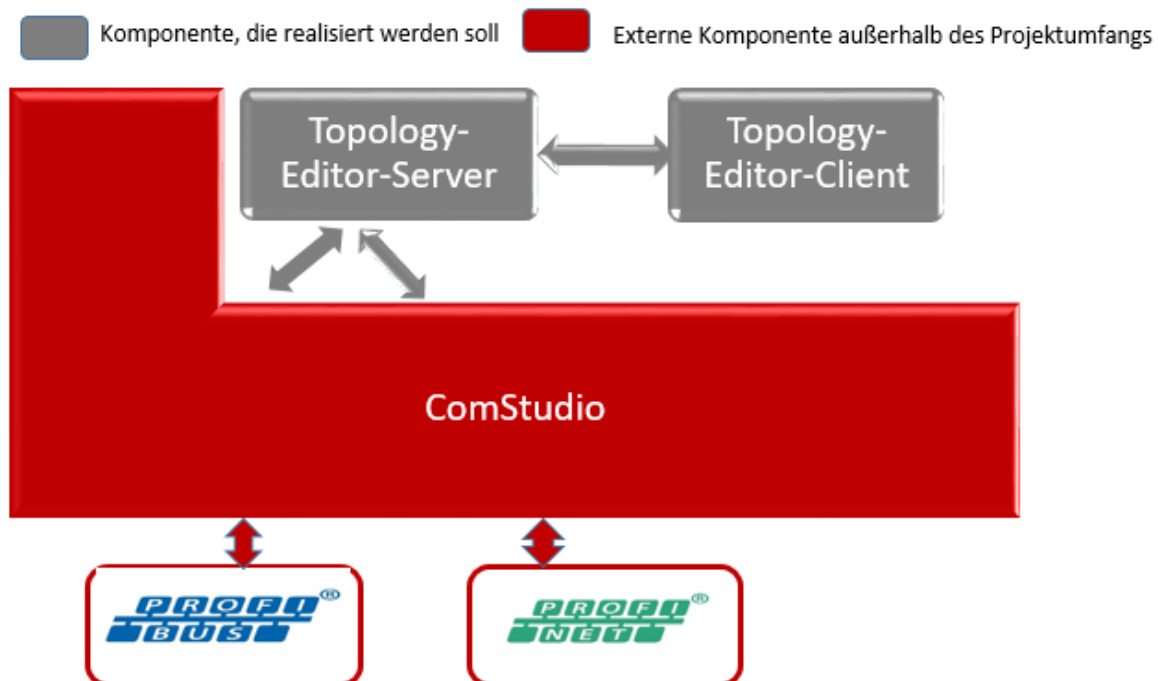


Abbildung 14: Kommunikation zwischen Feldbussystemen, ComStudio und Topology-Editor-Komponente

6.3 ComStudio-Szenario, User Interface

Abbildung 15 zeigt das Aussehen und die Funktionsweise der Benutzeroberflächenkomponente, wenn sie im ComStudio verwendet werden. Die generelle Idee für das Layout stammt von Kapitel 3.

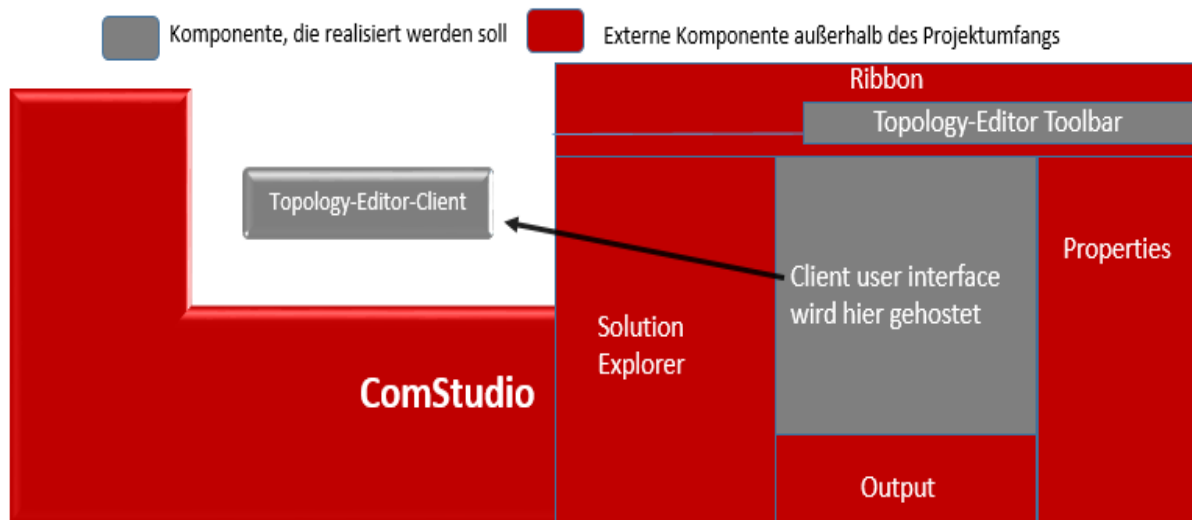


Abbildung 15: Das Layout für die Benutzeroberflächenkomponente

Die zu realisierende Komponenten sind in grau gekennzeichnet. Sie müssen die Größenänderung unterstützen. Das Topology-Editor Toolbar muss in Ribbon-Bereich eingebettet werden. In den Folgenden werden diese Komponenten in Details beschrieben:

- Topology-Editor Toolbar – umfasst eine Reihe von Befehlen wie Speichern, Laden, Zoomen, die direkt unter der vorgegebenen Ribbon-Leiste angeordnet sind. Er ermöglicht dem Anwender spezifische Aufgaben bezüglich der gewählten Topologie einfach zu erledigen. Um Ihre Funktionalitäten umzusetzen, müssen Entwickler die vordefinierte Klassenschnittstelle implementieren.
- Client User Interface - dieses Benutzersteuerelement wird verwendet, um dem Benutzer alle wichtigen Funktionen anzubieten. Das Benutzersteuerelement arbeitet sowohl offline (keine Verbindung zum herstellerspezifischen Gerät) als auch online.

6.4 Schnittstellen zwischen Komponenten

Zusätzlich zu den oben beschriebenen allgemeinen Einbettungsszenarien sollen folgende Anforderungen bezüglich der Kommunikation zwischen beteiligten Kompetenten durch das Realisierungskonzept erfüllt sein. Für die Interaktion zwischen den Komponenten ist eine Reihe von Schnittstellen erforderlich.



Abbildung 16: Topology-Editor Schnittstellen

Verantwortlichkeiten des Interface_DtmWebApi

- Steuerung der Geschäftslogik von Topology-Editor-Server durch die Einbettungsanwendung beispielsweise Lesen, Importieren, Exportieren, Löschen, Schreiben und Überwachen der Topologie
- Kommunikation mit feldbusspezifischen Geräten (Kommunikationszugriff wird von der Host-Applikation bereitgestellt). Die direkte Verwendung einer FDT-Kommunikationsschnittstelle wie IFdtCommunication (FDT Group AISBL, 2012) durch die Geschäftslogik ist nicht vorgesehen
- Persistenz des Instanz-Datasets (die Host-Applikation ComStudio hat Zugriff darauf per Referenz)
- Kann auch für die Interaktion zwischen der Geschäftslogik von Server und hersteller-spezifischen DTMs im FDT-Projekt verwendet werden

Verantwortlichkeiten der Topology-Editor REST-API

- Durch die Standardmethoden, die in Abschnitt 5.5 erläutert sind, biete diese Schnittstelle einen einheitlichen Zugriff auf Resources (Informationseinheiten)
- Da REST ressourcenorientiert ist, ist jede Topology-Editor-Client-Anfrage auf Resources bezogen
- Ihr zustandslosen Design bietet deutlich mehr Flexibilität beim Herstellen der Kommunikation zwischen Komponenten
- Als Format für Datenaustausch zwischen den beiden Topology-Editor-Komponenten wurde JSON gewählt. Es ist in den Komponenten einfach zu parsen und zu generieren.

6.5 Datendarstellung (Prozessdaten, Bedienungsdaten)

Der Hauptanwendungsfall für Topology-Editor ist die Offline/Online-Darstellung der Netzwerkstruktur und der gerätspezifischen Informationen in einer Browser-Komponente. Abbildung 17 zeigt, dass das Topology-Editor mit zwei verschiedenen Informationsquellen arbeiten kann:

- offline, d. h., Informationen ohne Verbindung zum Zielnetzwerk, konfiguriert oder geladen (siehe Persistenz) vom Nutzer.
- online, d. h., Informationen die nur verfügbar stehen, wenn eine Kommunikation mit dem physikalischen Feldbusgerät möglich ist. Diese Informationen nennt man Online-Topologie-Daten, sie werden durch einen Scan-Mechanismus abgefragt und können als Cache für die Daten der physikalischen Topologie betrachtet werden. Darüber hinaus sind diese Topologie-Daten immer synchron mit den Daten in der physikalischen Topologie.

Der Topology-Editor-Client verwendet immer die gesendeten JSON-Dateien, um eine Topologie protokollneutral darzustellen, zusätzlich kann der Client für den Datenzugriff eine Kopie der Daten haben und an diesen Daten arbeiten, bis Änderungen vom Anwender vorgenommen werden.

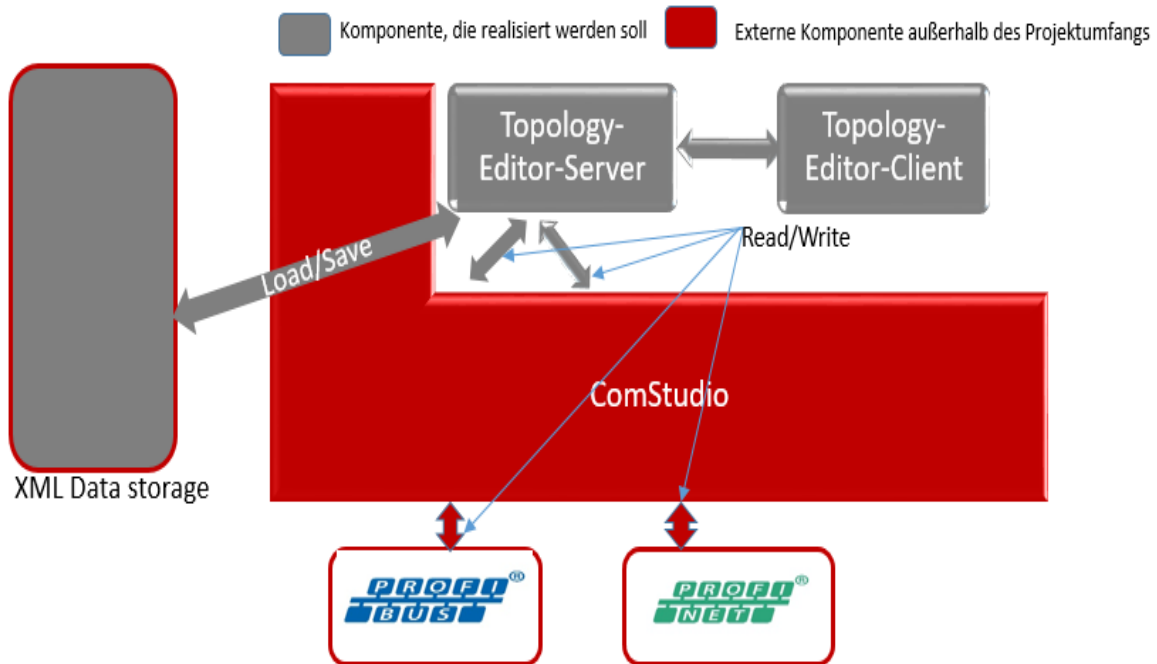


Abbildung 17: Datenhandhabungskonzept für Topology-Editor

Topology-Editor, JSON-Data Schema

Hier werden die Basisschemata, die bei Client-Anfragen angewendet werden, erläutert. Topology-Editor definiert Basisschemata, um die zu transportierenden Nachrichten zu beschreiben. Diese Schemata basieren auf JSON und ihre Struktur. Die Parameter und das Verhalten werden von der REST-API definiert. Die zu unterstützenden Dienste sind:

- CREATE: CreateDevice(), CreateLink()
- READ: GetDevices(), GetLinks()
- UPDATE: UpdateDevice(), UpdateLink()
- DELETE: DeleteDevice(), DeleteLink()

Abbildung 18 zeigt ein Beispiel von JSON-Datei, ein Client fordert gleichzeitig die Liste der Geräte- und dazugehörigen Verbindungsinformation. Die Server antwortet parallel mit zwei Dateien. Die generischen JSON-Dateien enthalten 2 Knoten und eine Kante: jeder Knoten repräsentiert ein Gerät und jede Kante eine spezifische Verbindung.

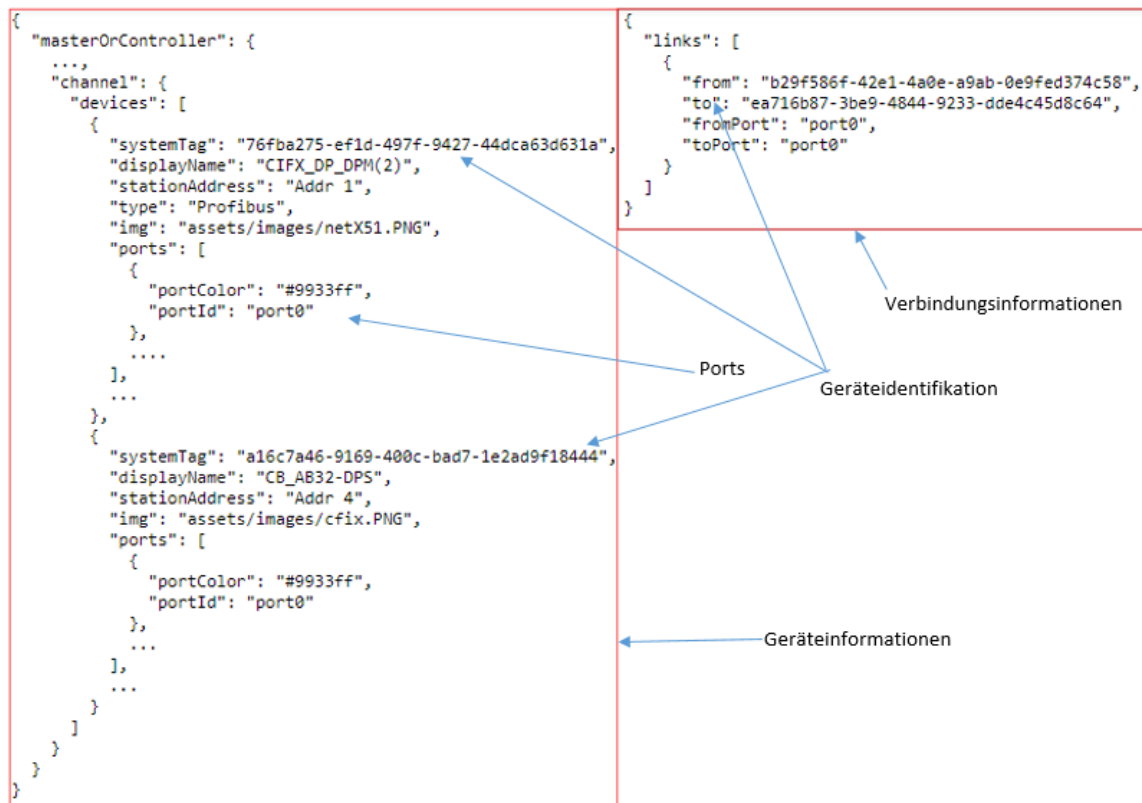


Abbildung 18: Die Output JSON-Dateien für die Mini-Topologie

„links“

„links“ ist ein konzeptioneller Schlüssel, der Eigenschaften enthält, die Teil die Anforderungsschlüssel sind, die bei Lesen, Schreiben und Aktualisieren von Verbindungsinformationen verwendet werden. Diese Eigenschaften werden hier zur Vereinfachung getrennt beschrieben

| Schlüssel | Beschreibung |
|------------------|---|
| from | Gibt Systemtag des Quell-Geräts an |
| to | Gibt Systemtag des Ziel-Geräts an |
| fromPort | Portkennung des Quell-Gerätes |
| toPort | Portkennung des Ziel-Gerätes |
| systemTag | Bietet eine intuitive Zuordnung zwischen der dargestellten Netzwerktopologie und einer Verbindung. Diese Eigenschaft ist optional |

„devices“

„devices“ ist ein konzeptioneller Schlüssel, der Eigenschaften enthält, die Teil die Anforderungsschlüssel sind, die bei Lesen, Schreiben und Aktualisieren von Geräteinformationen verwendet werden. Diese Eigenschaften werden hier auch zur Vereinfachung getrennt beschrieben:

| Schlüssel | Description |
|-----------------------|--|
| systemTag | Wird zur eindeutigen Identifizierung eines Geräts verwendet. In einem fehlerfreien Netzwerk besitzen alle Geräte ein systemTag. Der Wert dieser Elementeigenschaft muss in einer Netzwerktopologie eindeutig sein. Um die Eindeutigkeit zu gewährleisten ist dieser Wert als GUID ³³ gegeben. |
| displayName | Gibt den Namen des spezifischen Geräts an. Wenn diese Elementeigenschaft leer ist oder fehlt, ist der systemTag ein vollständig qualifizierter Name des Geräts. |
| stationAddress | Gibt die Stationsadresse eines spezifischen Geräts an |
| type | Gibt die unterstützten bzw. erforderlichen Protokolle eines spezifischen Geräts an. |
| img | Gibt das grafische Symbol (Icon, Bitmap) eines Geräts an. |
| ports | Gibt alle Ports eines spezifischen Geräts an. Sie ist einer der wichtigsten Elementeigenschaften der Device, diese darf nicht leer oder fehlerhaft sein und wird durch ein Array repräsentiert |

6.6 Zukunftsaspekte (evtl. Erweiterungen)

In diesem Abschnitt werden die Zukunftstrends für das Plugin zusammengefasst. Dabei liegt der Schwerpunkt auf der Performancesteigerung auf der sicheren und effizienten Benutzbarkeit des Plugins:

- **Übertragungssicherheit:** das Plugin soll bei Übertragung der Informationseinheiten eine Sicherheit gewährleisten. Denkbar wäre hier zum Beispiel die Implementierung der REST-API mit HTTPS anstatt HTTP
- **Echtzeitverhalten:** Der Topology-Editor soll die Veränderungen am physikalischen Netzwerk (Änderung der Geräteverkabelung, Geräteausfall, Kommunikationsstörung) innerhalb einer vorbestimmten Zeitspanne erkennen und Visualisieren.
- **Austauschformat:** es soll verschiedene Datenformate unterstützen, beispielsweise XML- bzw. HTML, so kann ein Topology-Editor-Client eine Ressource explizit anfordern.

³³ GUID steht für Globally Unique Identifier (Microsoft, 2018)

- **Logische Struktur:** es soll ein zusätzliches Softwaremodul implementiert werden, um die logische Topologie (siehe Abschnitt 2.1) aller protokollneutralen Feldbusse darzustellen
- **Datenspeicherung:** es soll mit einer Datenbank realisiert werden, um die steigende Anzahl der Parameter des Geräts gerecht zu werden. Denkbar wäre es eine SQL³⁴-Datenbank als Beispiel
- **Scan-Mechanismus**³⁵: es soll die Scan-Funktionalität unterstützen, damit die physikalischen Geräte-Informationen in Echtzeit zuverlässig angefordert werden können
- **Synchronisation:** es soll ein Synchronisationskonzept anbieten, um zwei Instanzen des Topology-Editors-Clients automatisch zu synchronisieren
- **Importfunktion:** es soll möglich sein, die dargestellte Topologie als Excel-Datei zu importieren
- **Exportfunktion:** es soll auch möglich sein, die Topologie-Informationen als eine Excel-Datei zu exportieren
- **User Experience:** neben der Übersichtlichkeit, die in der Anforderungsanalyse gelistet ist, soll es zusätzliche Funktionen anbieten wie
 - Gruppierung der dargestellten Geräte,
 - Einblenden bzw. Ausblenden der Geräte,
 - Einblenden bzw. Ausblenden der Verbindungen,
 - Autosave-Funktion

6.7 Entwurf

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

6.7.1 Architektur des Topology-Editor

Die Architektur des Topology-Editor basiert auf die Architektur einer SPA³⁶, denn der SPA-Ansatz reduziert die Zeit, die von der Anwendung für die Reaktion auf Benutzeraktion benötigt wird, was eine mehr flüssige Erfahrung ermöglicht.

In herkömmlichen Webanwendungen verarbeitet der Server die Anforderung und sendet Sie zum Browser als Reaktion auf die neue Aktionsanforderung von Client, jedoch bei SPA muss der Browser nur die Bereiche auf der Seite 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 präsent sind. Der Mix der eingesetzten Basistechnologien (ASP.NET:Web-API, JavaScript-Framework Angular 5 (Abschnitt 6.1) und CSS-Framework Bootstrap (Abschnitt 6.2)) vereinfacht die Umsetzung des Topology-Editors als SPA.

³⁴ SQL steht für Structured Query Language

³⁵ Scan-Mechanismus ist nicht Teil des Projektumfangs, prinzipiell ist ein Mechanismus um ein physikalisches Netzwerk online zu scannen

³⁶ Single Page Application

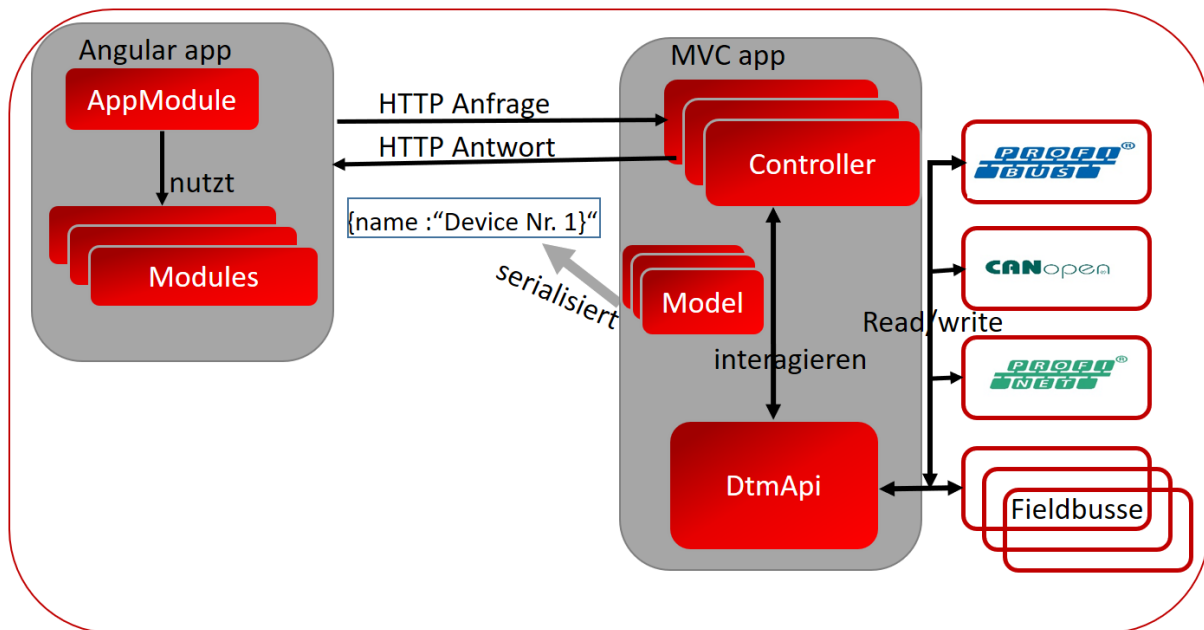


Abbildung 19: Diagramm zeigt den Grundlegenden Entwurf der Topology-Editor

Die Abbildung zeigt den Grundentwurf des Topology-Editor, der im Folgenden erklärt wird.

Der Client ist das Medium, das die Web-API nutzt (mobile App, Browser usw.). Hier wird der Client als Angular-App erstellt. Die Angular-App besteht aus 4 Hauptmodulen. Diese werden als TypeScript-Klassen dargestellt.

Die Modelle sind Objekte, die die Daten in der App darstellen. In diesem Fall sind drei Hauptklassen der erstellten Modelle als C#-Klassen umgesetzt, die auch als Plain Old C# Object (POCOs) bezeichnet werden.

Controller sind Objekte, die HTTP-Anforderungen verarbeiten und die HTTP-Antwort erstellen. 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.

Der Wrapper DtmApi ist die Schnittstelle für die feldbusprotokollabhängigen Parameter und die Einstellungen, die hier an der Stelle nicht betrachtet werden sondern als gegeben vorausgesetzt werden.

6.7.2 Entwurf der Serverkomponente

Die Serverkomponente des Topology-Editor wird als binäre, dynamisch verknüpfte Bibliothek geliefert. Abbildung 20 zeigt die wesentlichen Bibliotheken der Servers. Die grauen Blöcke sind die Bibliotheken (siehe Abschnitt 7.3.1 für mehr Information), die zu realisieren sind. Dieser Entwurf dient nur als Überblick über die Entwicklung der Plugin-Anwendung. Der Entwickler kann diese Muster als Grundlage für zukünftige Arbeiten verwenden.

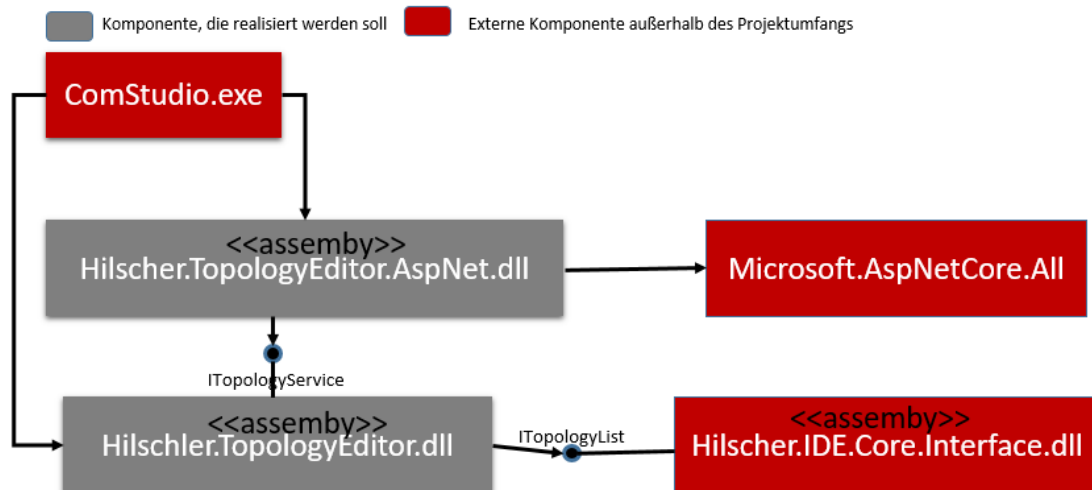


Abbildung 20: Komponentenübersicht für Topology-Editor-Server

6.7.3 Datenmodell

Aus Gründen der Übersicht wird in diesem Abschnitt nur auf die für die Plugin besonderes relevanten Entitäten der Topology-Editor eingegangen (Abbildung 21). Eine vollständige Übersicht befindet sich im angehängte CD.

Aus den funktionalen Anforderungen ergibt sich direkt das Domänenmodell.

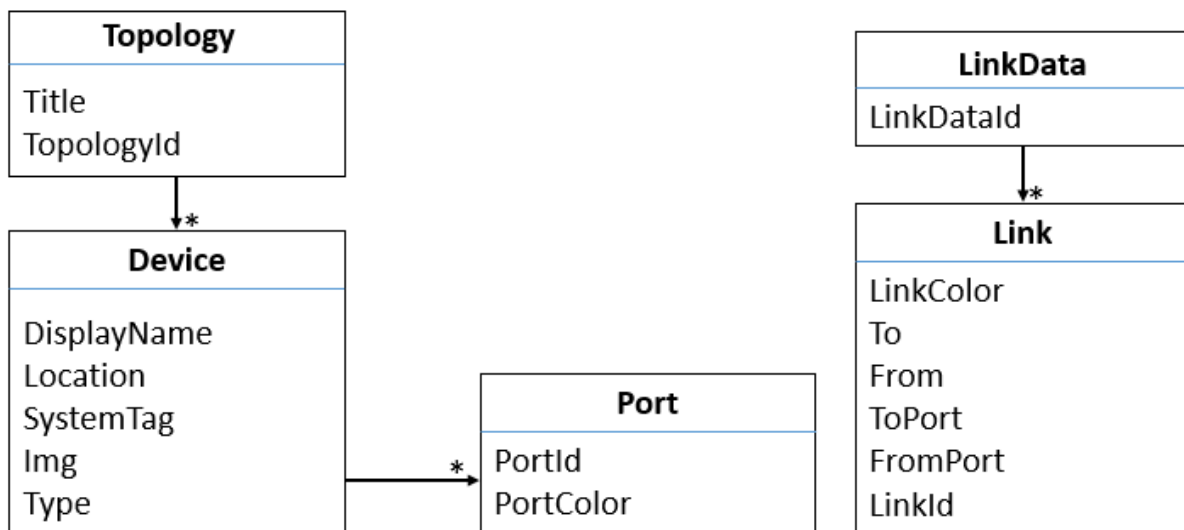


Abbildung 21: Domänenmodell der Elemente des Topology-Editor

Die 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.

Jedes *Device* kann wiederum mehrere Ports besitzen; Ein Port entspricht den Zugangspunkt eines *Device* und wird von der Entität *Link* verwendet, um die Verbindung herzustellen. Die Entität *LinkData* ist verantwortlich, die Verbindungen einer Topologie aufzulisten und die dazugehörige Topologie zu identifizieren.

6.7.4 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 bleiben 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.

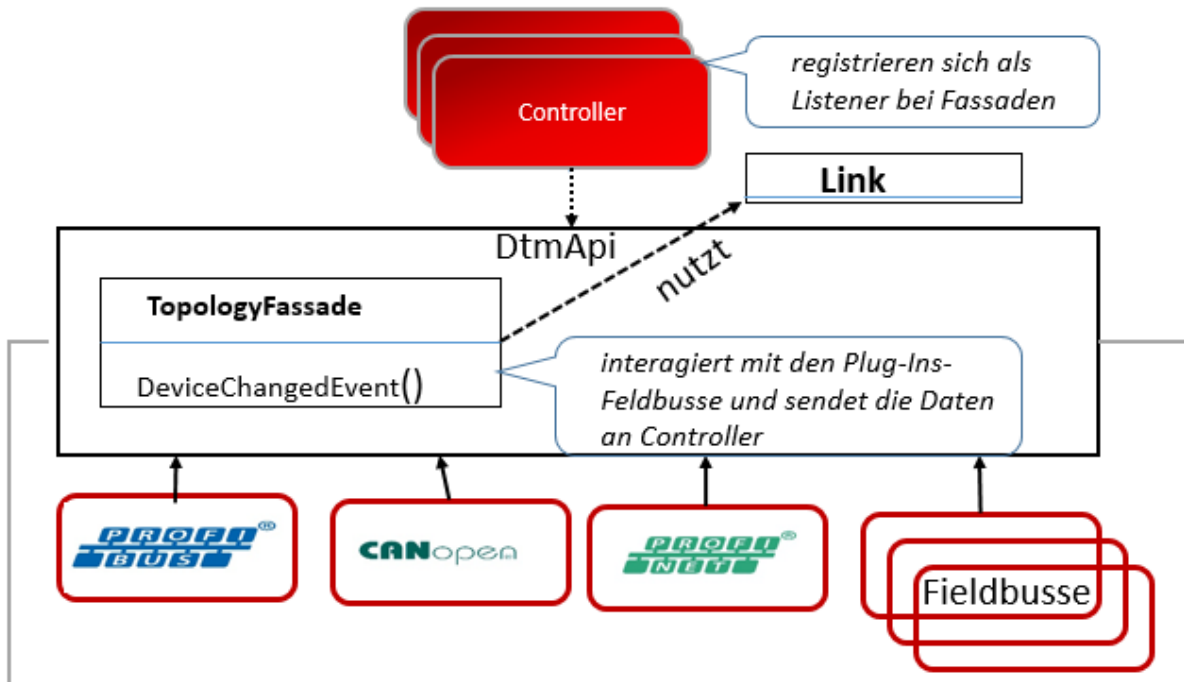


Abbildung 22: Klassendiagramm eines Beispiel für Fassaden mit feldbusabhängigen Protokollen

Abbildung 22 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 einer Methode gemacht (schwarz und fett), um die wichtige Rolle der Wrapper zu erklären. Die Plugins-Feldbusse sind Runde Rechtecke mit rote Rahmen.

7 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.

7.1 Vorgehensweise

Beim Unternehmen Hilscher ist das Scrum³⁷-Vorgehensmodell in der Abteilung Group User Interface für die Durchführung von Softwareprojekten vorgeschrieben. Da es sich um ein Plugin handelt, wird ein iterativ inkrementelles Vorgehensmodell eingesetzt. Dadurch können die implementierten Module schrittweise in die bestehende Software integriert werden.

7.2 Verwendete Werkzeuge

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

7.3 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 7.3.3 darauf eingegangen, wie verschiedene Geräte- und Verbindungsinformationen übertragen werden.

Anschließend wird in Kapitel 7.3.4 gezeigt, wie anhand dieser Informationen eine Topologie aktualisiert wird und geeignete Daten gefunden werden. Danach wird betrachtet, wie mit Hilfe der gefundenen Daten die Topologie erstellt wird. Abschließend wird die grafische Umsetzung der Darstellung und die Auswertung erläutert.

³⁷ ist die Bezeichnung für ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung.

7.3.1 Realisierung der Server-Applikation

Der Server ist ASP.Net Core basiert, mit der Vorlage Web-API. Er besteht aus drei wesentlichen Komponenten nämlich Hilscher.TopologyEditor, Hilscher.TopologyEditor.AspNet und die Hilscher.Web.API. Diese drei Komponente unterstützen die Software ComStudio dabei, alle Netzwerk-Protokolle für Web-Services mithilfe des Model-View-Controller-Musters zu kommunizieren. Sie wurden jedoch momentan hauptsächlich für PROFINET IRT entwickelt, die aus einzelnen, funktional vollständigen Teilen bestehen, die zusammen eine einzelne integrierte Schnittstelle bilden. Die Topology-Editor-Library beschleunigt die Entwicklung von Anwendungen mit bewährten Entwurfsmustern.

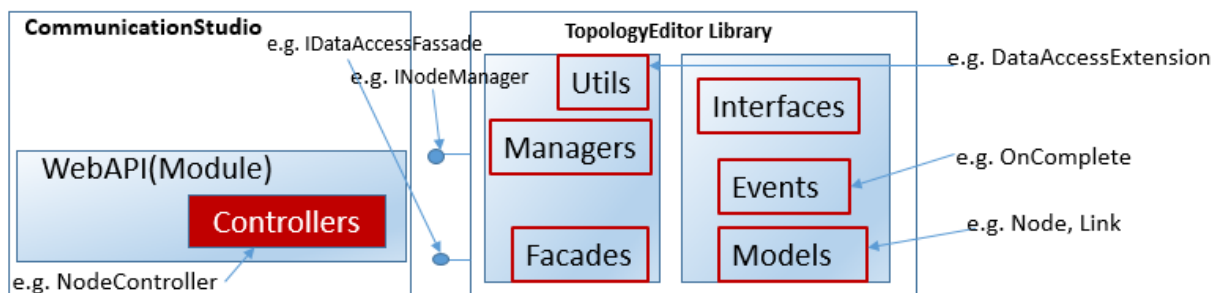


Abbildung 23: Zusammengesetzte Anwendungsarchitektur mit häufig verwendeten Paketen

Hilscher.TopologyEditor enthält die Kernfunktionalität von Topology-Editor 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 beispielsweise I-DataAccess, 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 Topology-Editor Library zu verwenden. Zu diesen Komponenten gehört die tatsächliche Implementierung von allen Schnittstellen von Interfaces und vieler Hilfsklassen

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

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

7.3.2 Realisierung der Client-Applikation

Wie bereits erwähnt, haben die benutzten Technologien, die zum Teil nach der Evaluation der JavaScript-Frameworks gewählt wurden und zum Teil durch die Problemstellung vorgegeben waren, beeinflussen sehr stark das Endprodukt und Entwicklung hinsichtlich der Client-Applikation.

Um einen Einblick in das Software-Modul Client-Topology-Editor selbst zu geben und aufzuzeigen wird das Modul-Diagramm mit Klassen von Client-Topology-Editor besprochen und dabei eine Übersicht über die Rolle von JSON-Dateien als Informationsquelle von Topology-Editor gegeben.

Zur Veranschaulichung der eigens für Client-Topology-Editor und der aus Angular-Framework und GoJs-Bibliothek verwendeten Klassen wurde das Diagramm in Abbildung 24 erstellt. Es zeigt das AppModule in der Mitte, um das sich alle Klassen von Client Topology Editor anordnen. Als Einstieg in das Diagramm können die Komponente-Klassen und Module-Klassen angesehen werden. Angular-Komponenten repräsentieren jeweils eine Ansicht in der App. Sicherlich startet man die Client-App als User nicht in irgendeiner Ansicht, sondern mit der Ansicht zum Editor, der „Topology Editor.html“.

Wie bereits angedeutet, wird eine Ansicht einer Angular-App zunächst als Component implementiert, dem man ein Template und ein CSS-File zuweist. Sobald das System eine Ansicht anzeigen soll, wird der Hauptteil der Funktionalität zu entsprechenden Service delegieren, da in Service die Hauptteile der Funktionalitäten umgesetzt wurden. Dadurch werden die Anwendungslogik und Präsentationlogik getrennt.

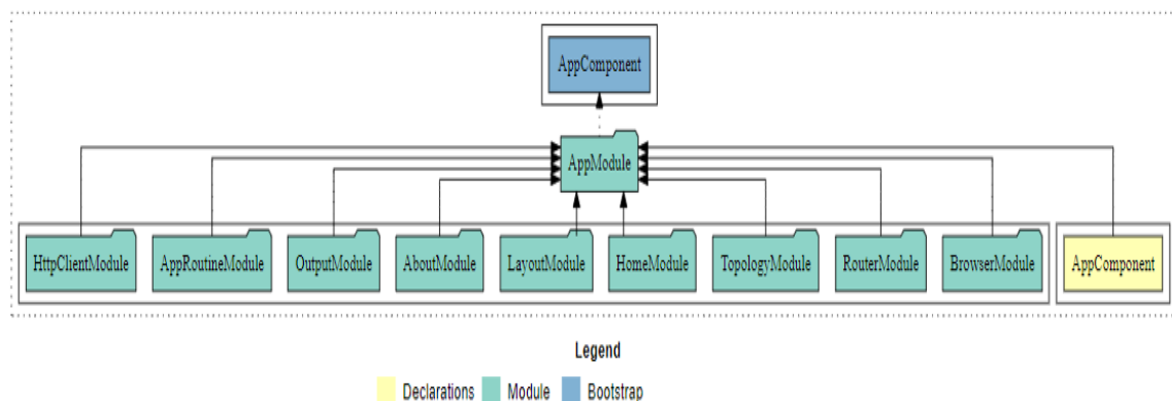


Abbildung 24: Vereinfachtes Diagramm für die Umsetzung der clientseitigen Anwendung

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

Manche Anweisungen, wie zum Beispiel http-Anfragen, können mitunter zu lange dauern und den UI-Thread blockieren. Um diese zu umgehen, müssen diese Funktionen in einem vordefinierten Verfahren von Angular-Framework ausgelagert werden. Dieses Verfahren wurde zum Senden und Empfangen der Topologie-Informationen im System im Rahmen des CRUD³⁸-Operators in NET.Core 2.0 (Microsoft, 2018) umgesetzt.

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

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

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

Für die Kommunikation mit den beiden JSON-Dateien sind noch die Klassen „DeviceService.ts“ und „LinkService.ts“ verantwortlich. Sie verfügen über einem großen Satz an Methoden, um entsprechende JSON-Dateien im System zu manipulieren, weshalb sie eine zentrale Rolle in den Modulen spielen. Die beiden Klassen sind im Singleton-Pattern³⁹ erstellt worden und wurden in entsprechenden Komponenten integriert.

7.3.3 Systemvorbereitung und Erstellen eines FDT-Projekts

Um das Plugin zu nutzen, muss der Nutzer vor allem die Software ComStudio starten. Dabei wird der Server für das Topology-Editor 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.

³⁸ CRUD steht für Create, Read, Update, Delete

³⁹ Ein Singleton-Pattern sorgt dafür, dass es von einer Klasse nur eine einzige Instanz gibt und diese global zugänglich ist

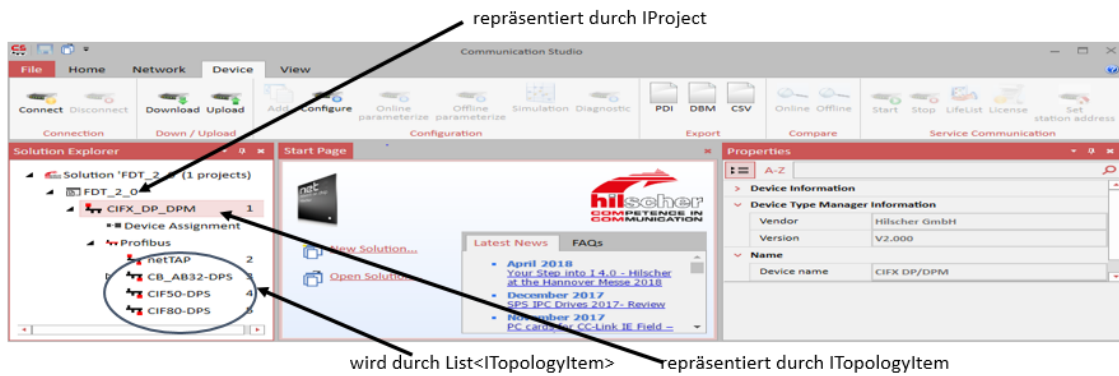


Abbildung 25: Erstellen eines FDT-Projekt

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 Topology-Editor 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.

7.3.4 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 aber für das gesamte Projekt da es nicht Teil der Aufgabe ist.

Der Nutzer kann entweder das Topology-Editor-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.

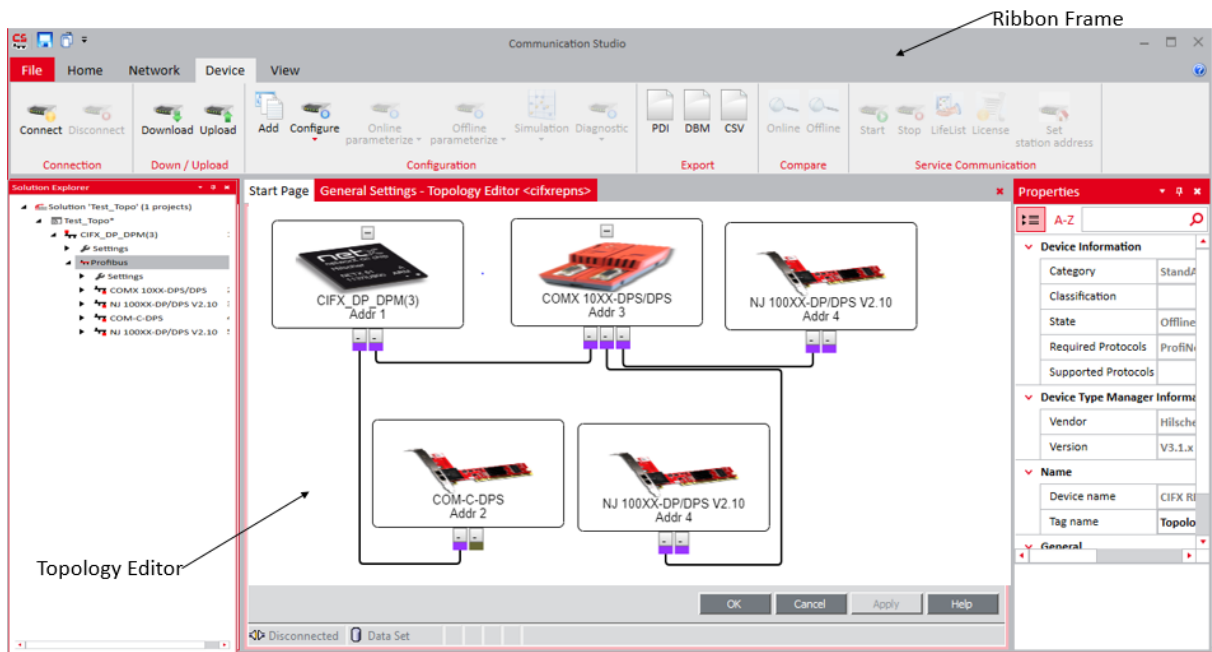


Abbildung 26: ComStudio mit Topology Editor im Einsatz

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

8 Zusammenfassung und Ausblick

In diesem Kapitel werden die vorgestellten Inhalte zusammengefasst und Ansatzpunkte für die zukünftige Weiterentwicklung des Topology-Editors in einem Ausblick gegeben.

8.1 Zusammenfassung

Nach einer Übersicht über alle Topologie-Arten, wurde der Ist-Zustand hinsichtlich der Com-Studio dargestellt und die Anforderungen für das Plugin wurden analysiert. Dabei konnten mehrere Schwachstellen bezüglich des Konfigurationprozesses mit dem bestehenden Konfigurator festgestellt werden. Die Analysen der logischen Verbindungen und des Echtzeitverhaltens waren nicht Bestandteile dieser Arbeit.

Der Aufbau des Topology-Editors und die grafische Repräsentation der Geräte samt deren semantischen Informationen erlaubt es dem Inbetriebnehmer, Hilscher-Kunden wie Hilscher-Ingenieuren, einfach und mit wenig Aufwand Netzwerke zu konfigurieren.

Die Darstellung der Verbindungen zwischen den Geräten bietet eine 1:1 Sicht auf die physikalische Verkabelung / Vernetzung der Feldbusgeräte in einem Netzwerk. Dadurch kann der Anwender die Beziehungen zwischen den Kommunikationsteilnehmern schnell erfassen. So einfach und sicher zu bedienendes Plugin-Modul Topologie-Editor ist vollkommen geeignet für die Einsparung der Engineering-Zeit und somit für eine Kostenreduktion bei der Inbetriebnahme der Anlagen.

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- und Web-Anwendungen aus einer gemeinsamen Codebasis lässt sich mit dem JavaScript-Framework Angular 2 umsetzen. Die Realisierung mit diesem Framework erlaubt es im Vergleich mit den anderen Optionen aus Kapitel 5 den größten Grad der Wiederverwendung des Quellcodes für die Plattformen und Webbrowser.

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 Konzepten 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 6.1 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.

8.2 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*. Das Problem wird mit der Zeit immer größer, da die Anforderungen an Netzwerkenperformance und Gerätekomplexität steigen stetig. Die Übertragungsgeschwindigkeit der Daten wurde nicht betrachtet, sie soll im weiteren Ausbauschnitt des Plugins adressiert werden.

Das Einführen der Single Page App (Siehe Abschnitt 7.1) war ein erster richtiger Schritt in die richtige Richtung, um die Übertragungszeit zu reduzieren. Sie spart viele tausende Abfragen der Daten pro erstellter Netzwerk-Topologie, was bisherigen zeitlichen Aufwand ungefähr halbiert hat.

Bei dem Topology-Engineering-Tool geht es vor allem darum, die Verknüpfung der Geräte eines industriellen Netzwerks darzustellen. Eine sinnvolle, künftige Erweiterungsmöglichkeit wäre die logische Darstellung des Datenflusses im Netzwerk. Ebenfalls könnten die Anzeigen der unterlagerten Netzwerke über die entsprechenden Kommunikationsinfrastrukturgeräte (Gateways) flexibilisiert werden.

Man könnte auch die Linien in der Topologie kontextuell verbessern, indem man sie farblich je nach Protokoll kennzeichnet.

Eine andere umsetzbare Erweiterung für den Topology-Editor wäre die Übertragungssicherheit, was in der industriellen Kommunikation einer großen Rolle spielt.

Der Topology-Editor kann auch als interaktives Hilfsmittel beim Lernen der verschiedenen Topologie-Arten eingesetzt werden. Man könnte das Plugin um die Möglichkeit erweitern, dass ein Benutzer die Topologie-Arten, siehe Abschnitt 2.2, umschalten kann.

In Zukunft wird es immer mehr darum gehen, die technischen Möglichkeiten sinnvoll miteinander zu verknüpfen um die Anwendung für den Endnutzer intuitiv und adaptierbar zu gestalten und ihm die Informationen zur Verfügung zu stellen, die er benötigt.

Anhang: Use Cases

| Use Case | Topologie erstellen und anzeigen |
|--------------------|--|
| ID | #01 |
| Beschreibung | Der Nutzer kann zu jeder Zeit eine Topologie erstellen sobald der Topologie-Editor-Server in ComStudio läuft, d.h., ComStudio starten, neues Feldbus-Projekt anlegen, ein Master-Device einfügen, ein oder mehrere Slave-Devices einfügen. Eine Topologie anzuzeigen bedeutet, einen Graph im Topologie-Editor mit den entsprechenden Geräten anzuzeigen. |
| 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 26 |

Tabelle 4: Use Case #1

| Use Case | Topologie bearbeiten |
|--------------------|---|
| ID | #02 |
| Beschreibung | Der Nutzer kann eine Topologie fast zu jeder Zeit bearbeiten, d.h. den Namen eines Geräts ändern, die Verbindungen tauschen oder sogar ein neues Gerät in der bestehenden Topologie einfügen. Die Topologie kann nur geändert werden, wenn ComStudio nicht in Online-Modus ist. |
| Beteiligte Akteure | Inbetriebnehmer / Kunde |
| Vorbedingung | ComStudio und Topologie-Editor-Server laufen und in ComStudio steht mindestens ein FDT-Projekt zur Verfügung |
| Trigger | Der Nutzer klickt auf „Topology-Editor“ |
| Nachbedingung | Alle Änderungen an den Daten der Topologie werden zu Topology-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 sie häufig genutzt wird |

Tabelle 5: Use Case #2

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

Tabelle 6: Use Case #3

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

Tabelle 7: Use Case #4

Literatur

- AG, S. (01 2018). *SIEMENS*. Abgerufen am 01. 03 2018 von <http://w3.siemens.com/mcms/automation/de/industrielle-kommunikation/profibus/Seiten/Default.aspx>
- Alexandroff, Paul, Hopf, Heinz. (1945). *Topologie*. Digitalisierungsprojekts Springer Book Archives.
- cmb-systeme. (05 2004). *CMB SYSTEME*. Abgerufen am 19. 01 2018 von <https://www.cmb-systeme.de/technikwissen/was-bedeutet-ethernet>
- Conrads, D. D. (2014). Telekommunikation. In *Grundlagen, Verfahren, Netze, Ausgabe 5* (S. 13-15). Wiesbaden: Vieweg & Sohn Verlag.
- Facebook Inc. (2018). *React*. Abgerufen am 05. 05 2018 von <https://reactjs.org/>
- Facebook Inc. (2018). *React Native*. (React) Abgerufen am 05. 05 2018 von <https://facebook.github.io/react-native/>
- Facebook OpenSource. (2018). *React*. (Facebook Inc.) Abgerufen am 01. 03 2018 von <https://reactjs.org/docs/introducing-jsx.html>
- Facebook/React. (2018). *Github*. Abgerufen am 05. 05 2018 von <https://github.com/facebook/react/releases/tag/v16.3.2>
- FDT Group AISBL. (2012). FDT2.0 Technical Specification, Version 1.00. Document No. 0001-0008-000. In *FDT2_Specification v1.00.pdf*.
- Google. (01 2010). *Angular- Resolve*. Abgerufen am 11. 02 2018 von <https://angular.io/api/router/Resolve>
- Google. (2018). *Angular*. (Google) Abgerufen am 05. 05 2018 von <https://angular.io/guide/architecture-services>
- Gottfried Vossen, K.-U. W. (2016). *Grundkurs Theoretische Informatik*. Trier: Vieweg+Teubner Verlag .
- Heinrich Hippenmeyer, T. M. (2016). *Automatische Identifikation für Industrie 4.0*. Berlin, Heidelberg: Springer Vieweg.
- Heinrich Hübscher, H.-J. P. (2011). *IT-Handbuch*. Braunschweig: Westermann Schroedel Diesterweg Schöningh Winklers GmbH.
- Hilscher Gesellschaft für Systemautomation mbH. (02 2016). *Hilscher Gesellschaft für Systemautomation mbH*. (netX) Abgerufen am 02. 02 2018 von <https://www.hilscher.com/de/netx/einfuehrung/>
- Hilscher GmbH. (01. 10 2017). *Portal Hilscher*. Von <https://kb.hilscher.com/display/COMSTUDIO/2017-06-02+Topology+in+Communication+Studio> abgerufen
- HMS. (2015). (HMS Industrial Networks) Abgerufen am 05. 05 2018 von <http://www.feldbusse.de/Profinet/profinet.shtml>
- IT Wissen Info. (2001). *IT Wissen Info*. (2018) Abgerufen am 02. 02 2018 von <https://www.itwissen.info/Ethernet-Ethernet.html>

- Kersken, S. (2013). *IT-Handbuch für Fachinformatiker*. Bonn: Galileo Press.
- KUNBUS GmbH. (2016). *KUNBUS*. () Abgerufen am 01. 03 2018 von <https://www.kunbus.de/feldbus.html>
- LinkedIn. (2018). *LinkedIn*. Abgerufen am 05. 05 2018 von <https://www.linkedin.com/in/evanyou/de>
- MDN web docs. (01 2005). *Web API Referenz*. Abgerufen am 01. 03 2018 von <https://developer.mozilla.org/de/docs/Web/API>
- Microsoft . (02 2018). *Visual Studio-IDE*. Abgerufen am 04. 04 2018 von <https://www.visualstudio.com/de/vs/>
- Microsoft. (05 2018). <https://msdn.microsoft.com/de-de/library/bb979128.aspx>. Von MSDN: <https://msdn.microsoft.com/de-de/library/bb979128.aspx> abgerufen
- Microsoft. (01 2018). *mplementieren grundlegende CRUD-Funktionalität mit Entity Framework in ASP.NET MVC-Anwendung*. Abgerufen am 12. 04 2018 von <https://docs.microsoft.com/de-de/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-basic-crud-functionality-with-the-entity-framework-in-asp-net-mvc-application>
- Microsoft. (01 2018). *Visual Studio Code - Code Editing*. Abgerufen am 04. 04 2018 von <https://code.visualstudio.com>
- Microsoft Corporation. (2018). *Microsoft . (ASP.NET Core)* Abgerufen am 04 2018 von <https://docs.microsoft.com/de-de/aspnet/>
- Northwoods Software. (01. 01 1998). (GOJS) Abgerufen am 05. 04 2018 von <https://www.nwoods.com/store/p-82-gojs-oem.aspx>
- Northwoods Software. (01. 01 1998). *GoJS*. Abgerufen am 04. 04 2018 von <https://gojs.net/latest/index.html>
- Olbrich, A. (2003). *Netze — Protokolle — Spezifikationen*. Wiesbaden: Vieweg & Sohn.
- Plenk, P. D.-I. (2017). *Angewandte Netzwerktechnik kompakt*. Wiesbaden: Springer Fachmedien Wiesbaden GmbH.
- Popp, M. (2016). *Industrielle Kommunikation mit PROFINET*. Karlsruhe: ProfiBus Nutzerorganisation e.V.(PNO).
- Progress Software Corporation . (2018). www.progress.com. (Progress Software Corporation) Abgerufen am 05. 05 2018 von <https://www.progress.com/kendo-ui>
- René Simon, T. K. (2003). *Field Device Tool - FDT*. München : Oldenbourg Wissenschaftsverlag.
- Ringhand, Z. (2010). IT- Handbuch. In *IT-Systemelektroniker/-in Fachinformatiker/-in* (S. 314-315). München: Westermann.
- SAP. (01 2018). *UI5 Demo Kit - UI Development Toolkit for HTML5*. (UI5) Abgerufen am . 03 2018 von <https://sapui5.hana.ondemand.com>
- Schnabel, P. (2018). *Elektronik-Kompendium.de - Elektronik einfach und leicht verständlich*. Von <https://www.elektronik-kompendium.de/sites/net/0503281.htm> abgerufen
- Sommerville, I. (2007). *IT Informatik Software Engineering*. Pearson.
- Twitter. (01 2010). *GetBootstrap*. (Bootstrap) Abgerufen am 02 2018 von <https://getbootstrap.com>
- Vue.Js, G. (2018). *GitHub*. Abgerufen am 05. 05 2018 von <https://github.com/vuejs/vue/releases/tag/v2.5.16>
- w3schools.com. (01 1999). *W3.CSS Templates*. Abgerufen am 03. 03 2018 von https://www.w3schools.com/w3css/w3css_templates.asp
- Weebly. (01 2016). *netzwerk-topologien.weebly*. Abgerufen am 01. 03 2018 von <https://netzwerk-topologien.weebly.com/grad.html>
- You, E. (2018). *VueJS*. Abgerufen am 05. 05 2018 von <https://vuejs.org>

Index

A

ASP.NET Core 2.0..... 33

C

Communication..... 16, 17

Component..... 25

D

Device 7

E

Echtzeitverhalten..... 1

Ethernet-Technik..... 6

F

FDT 16

funktionalen Anforderungen 20

G

GET..... 33

GoJS..... 28

K

Kendo UI..... 27

M

Module 25

N

Netzwerk 6

Netzwerkprotokoll 6

Nichtfunktionale Anforderungen 22

O

OSI-Referenzmodell 7

P

Plugin..... 2

POST..... 33

PROFIBUS-FMS..... 14

Programmierkonzept..... 29

R

React 26

S

Software 2

Supervisor 7

T

Template 25

Topology-Editor..... 34, 45

U

Übertragungssicherheit 1

URL 32

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 Technologie |
| 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