

TECHNISCHE  
UNIVERSITÄT  
DRESDEN



---

Fakultät Elektrotechnik und Informationstechnik Institut für Prozessleittechnik

---

# PROJEKTIERUNG VON AUTOMATISIERUNGSSYSTEMEN

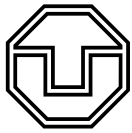
zum Thema

Entwicklung eines OPC-UA basierten Process Historian für  
modulare Anlagen

vorgelegt von Marc Becker, Erik Noak, Max Kirchner, Patrick Suwinski,  
Oliver Parczyk, Daniel Kluge  
im Studiengang Elektrotechnik, Informationssystemtechnik

Betreuer: Dr. rer. nat. Valentin Khaydarov  
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. habil. Leon Urbas  
Tag der Einreichung: 17. Mai 2021





Fehlt noch !!

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Betreuer: Dr. rer. nat. Valentin Khaydarov  
Hochschullehrer: Prof. Dr.-Ing. habil. Leon Urbas  
Tag der Einreichung: 17. Mai 2021

---

Bearbeiter: Marc Becker, Erik Noak, Max Kirchner, Patrick Suwinski,  
Oliver Parczyk, Daniel Kluge



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Aufgabenplanung . . . . .	4
<b>2</b>	<b>Grundlagen und Stand der Technik</b>	<b>7</b>
2.1	Open Platform Communication . . . . .	7
2.1.1	Entwicklung . . . . .	7
2.2	OPC - Unified Architecture . . . . .	7
2.2.1	Aufbau . . . . .	8
2.2.2	Sicherheit . . . . .	9
2.2.2.1	Übertragungssicherheit . . . . .	9
2.2.2.2	Zugriffssicherheit . . . . .	9
2.2.3	Anwendungsbereiche . . . . .	10
2.3	Module Type Package . . . . .	10
<b>3</b>	<b>Architektur</b>	<b>13</b>
3.1	Komponentendiagramm . . . . .	13
3.2	Klassendiagramm . . . . .	13
3.3	Sequenzdiagramm . . . . .	14
3.4	Anwendungsfalldiagramm . . . . .	17
3.5	Konfigurations-Schemata . . . . .	17
<b>4</b>	<b>Systemanforderungen</b>	<b>19</b>
<b>5</b>	<b>Bedienungsanleitung</b>	<b>21</b>
	<b>Literaturverzeichnis</b>	<b>23</b>



## Abbildungsverzeichnis

1.1	Datenstruktur . . . . .	2
1.2	Generelle Struktur der Architektur . . . . .	3
1.3	Meilensteine . . . . .	4
1.4	Timeline der Projektplanung . . . . .	6
2.1	OPC Netzstruktur aus Sicht des Servers . . . . .	8
2.2	OPC Netzstruktur aus Sicht des Clients . . . . .	9
2.3	Aufbau eines Module Type Package . . . . .	11
3.1	Komponentendiagramm . . . . .	13
3.2	Klassendiagramm . . . . .	14
3.3	Sequenzdiagramm . . . . .	16
3.4	Use-Case-Diagramm . . . . .	17
3.5	UPC UA-Config-Schema . . . . .	18
3.6	Program-Config-Schema . . . . .	18





# 1 Einleitung

In der Automatisierungstechnik werden modulare Anlagen immer häufiger verwendet, da es viele Vorteile bietet. Es ist möglich, Anlagen aus verschiedenen Komponenten zusammenzustellen und je nach Bedarf können einzelne Module einfach ausgetauscht bzw. erweitert werden. Diese Modularisierung stellt bei der Entwicklung neue Herausforderungen. Es müssen Schnittstellen implementiert werden, sogenannte Module Type Packages (MTP), so dass die Kommunikation und Funktionalität zwischen den einzelnen Process Equipment Assembly (PEA), auch Modul genannt, gewährleistet wird. Eine modulare Anlage benötigt einen Datenaustausch zwischen den einzelnen PEA, um als gesamte Einheit funktionieren zu können. Dabei unterscheidet man zwischen Engineering Daten, Live-Daten und historische Daten. Ziel dieses Projektes ist es, ein OPC UA-basiertes Tool zu entwickeln, Process Historian genannt, um den Prozess überwachen und auswerten zu können.

Ausgehend von der Triple-Store Datenbank wird die Information erhalten, welche Prozesswerte vorhanden sind. Diese werden in Form von Attributen wie NodeID, Name und Namespace in einer Konfigurationsdatei abgespeichert. Diese Datei legt zusätzlich die Art und Weise fest, wie Live-Daten abgefragt werden. Nach dem Subscriben bzw. Pollen der Prozessdaten über den OPC UA Client werden über einen Buffer die Daten in die InfluxDB hochgeladen und abgespeichert. Auf die abgespeicherten Daten wird mit der Auswertungsplattform Grafana zugegriffen, womit diese visualisiert werden.

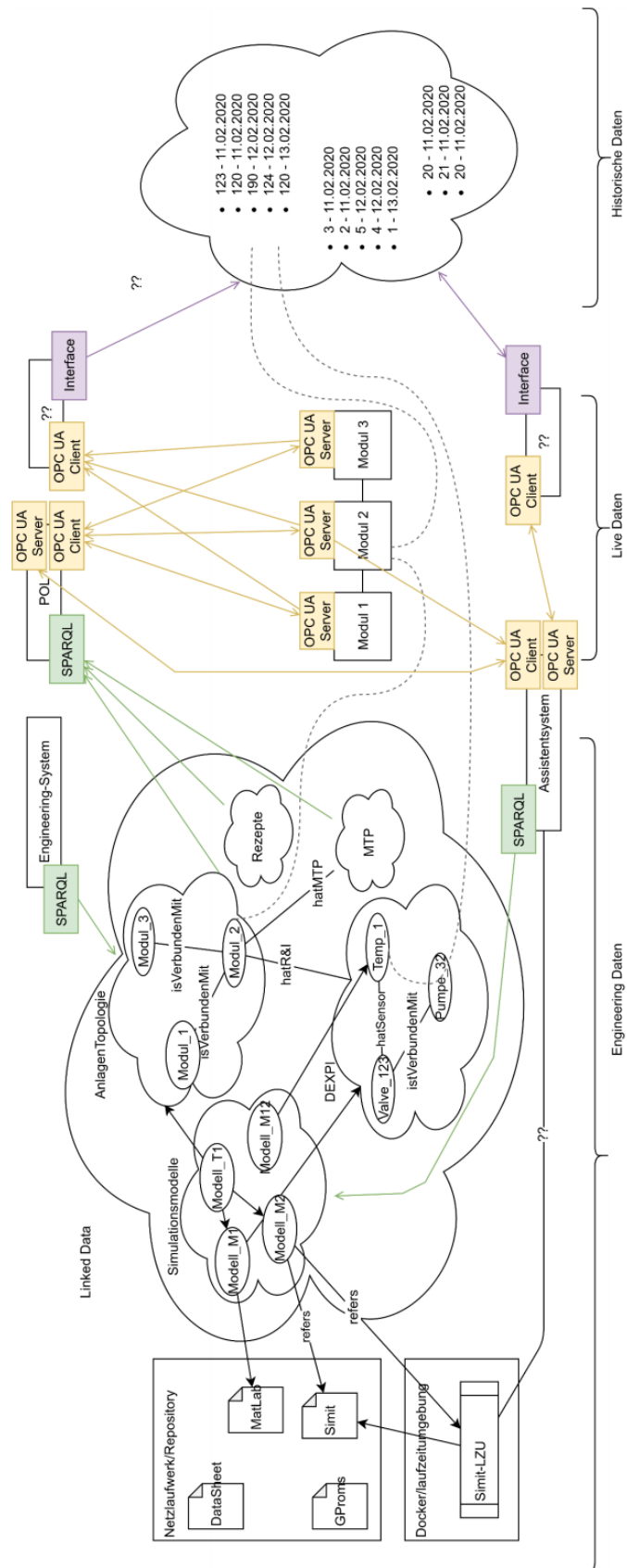
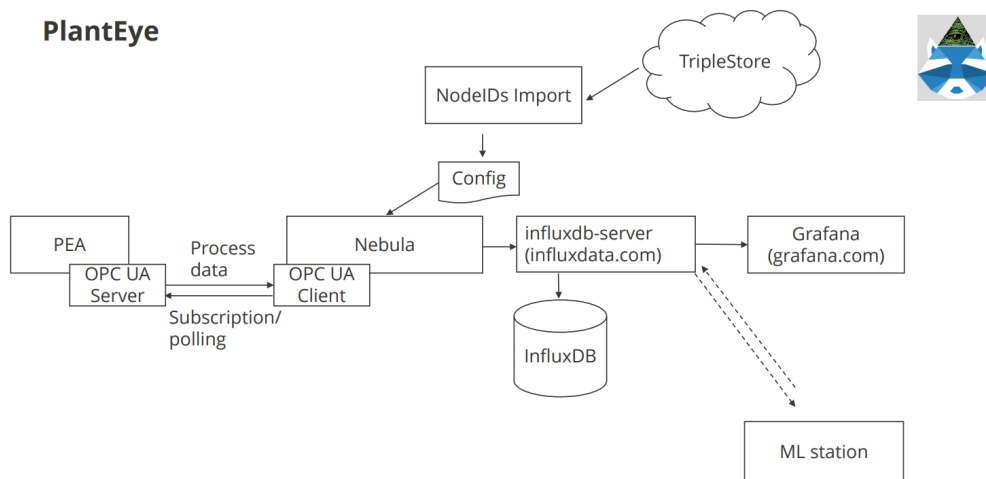


Abbildung 1.1: Datenstruktur



**Abbildung 1.2:** Generelle Struktur der Architektur

### 1.1 Aufgabenplanung

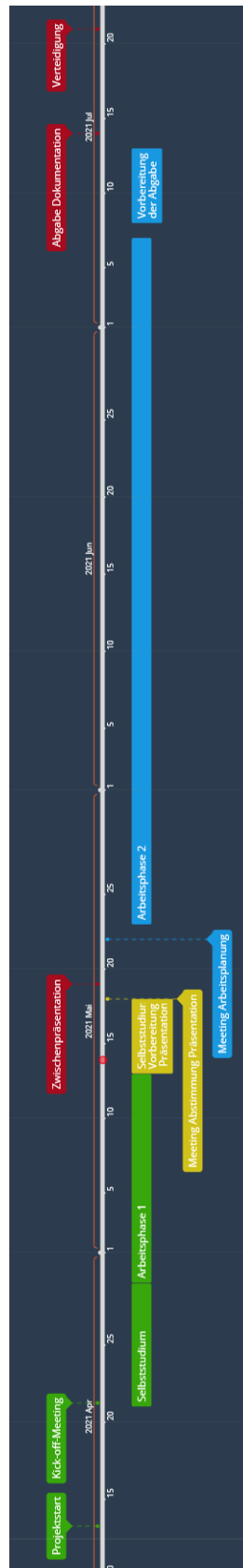
Das Ziel der Gruppe ist es, das Projekt parallel zu bearbeiten, weshalb es notwendig ist, über GitHub eine Versionskontrolle einzuführen. Für das erfolgreiche Abschließen des Projekts ist die Einführung von Etappenzielen, sogenannten Meilensteinen, sehr wichtig. Für die Planung der Zwischenziele wurde eine Tabelle erstellt, in welcher das Ziel mit einer Deadline festgehalten wird. Somit hat jedes Gruppenmitglied einen schnellen Überblick über die zu erledigenden Aufgaben und es ist ersichtlich, welcher Stand erreicht wurde. Aufgrund der Komplexität der Aufgabe wächst diese Tabelle zunehmend mit den wachsenden Erkenntnissen. Mit einer grünen Farbcodierung ist ersichtlich welche Meilensteine bereits abgeschlossen wurden. Gelb und Rot charakterisieren sehr wichtige Termine.

Meilenstein	Deadline	Verantwortlich
Start	13.04.2021	Alle
Kickoff	21.04.2021	Alle
Selbststudium	28.04.2021	Alle
UML-Klassendiagramm	12.05.2021	Daniel, Oliver
Problemanalyse, Arbeitsplan	12.05.2021	Max, Marc
Theorie zu MTP, OPC	12.05.2021	Patrick, Erik
Doku Architektur	12.05.2021	Daniel, Oliver
Folien Präsentation	18.05.2021	Alle
Zwischenpräsentation	19.05.2021	Alle
Dokumentation aller Ergebnisse	07.07.2021	t.b.a
Beschreibung Implementierung	07.07.2021	t.b.a
Dokumentation geeigneter Tests bzw. Validierungstests	07.07.2021	t.b.a
Kritische Diskussion der Arbeitsergebnisse	07.07.2021	t.b.a
Abgabe 1	14.07.2021	Alle
Verteidigung	21.07.2021	Alle

**Abbildung 1.3:** Meilensteine

In diesem Projekt hat Max Kirchner die Position des Projektmanagers inne und ist in erster Linie für die Planung als auch die Kommunikation mit dem Betreuer zuständig. Die Architektur wird von Daniel Kluge und Oliver Parczyk geplant. Das beinhaltet die Strukturierung des Projekts in Form von UML-Klassendiagrammen, Komponentendiagrammen und Use-Case-Diagrammen. Erik Noack, Patrick Suwinski und Marc Philipp Becker sind hauptsächlich für die Implementierung verantwortlich. Für eine erfolgreiche Umsetzung ist eine gute Zusammenarbeit zwischen den Gruppen not-

wendig, weshalb die Aufgaben auch gemeinsam gelöst werden.



**Abbildung 1.4:** Timeline der Projektplanung  
„Web-Tool Timeline der Projektplanung“, 04-05-2021

## 2 Grundlagen und Stand der Technik

### 2.1 Open Platform Communication

#### 2.1.1 Entwicklung

Open Platform Communications (*OPC*) ist ein im industriellen Umfeld weit verbreiteter und etablierter Kommunikationsstandard. Die erste Spezifikation *OPC Specification Version 1.0* (seltener Classic OPC) wurde 1996 veröffentlicht, wobei die Unternehmen Siemens, Fisher-Rosemount und Intellution den größten Einfluss auf die Entstehung hatten. („History OPC“, 10-05-2021) („Classic OPC“, 10-05-2021) Diese Unternehmen gründeten kurz nach der Veröffentlichung der *OPC Foundation*, der seitdem eine hohe Anzahl an Unternehmen beitraten und mittlerweile 817 Industriepartner umfasst. („OPC Foundation Members“, 10-05-2021) Classic OPC wurde im Laufe der Zeit durch weitere Spezifikationen wie *OPC Data Access* (OPC DA), *OPC Historical Data Access* (OPC HDA), *OPC Alarm and Events* (OPC AE oder OPC A&E) weiterentwickelt. Die aktuellste Spezifikation stellt dabei *OPC Unified Architecture* (OPC UA) dar. Der Classic OPC zeichnete sich durch eine Client-Server-Verbindung, eine stark variierenden Funktionalitätsunterstützung und einer schlechten Skalierung aus. Außerdem war eine Microsoft-Lizenz nötig. Jedoch ist Classic OPC die Grundlage für alle weiteren Spezifikationen. OPC DA umfasst einen wertebasierten Datenaustausch, einen Zeitstempel und bietet die Möglichkeit einer Aussage über die Qualität der gesendeten Daten. Als weitere Spezifikation definiert OPC HDA die Abfragemethoden (query), Analysen (analytics) und verfügt ebenfalls über die Angabe eines Zeitstempels. Ein umfassender Datenaustausch von Alarmen, Events, Variablenstatus und Statusmanagement wurde mit der Spezifikation von OPC AE möglich.

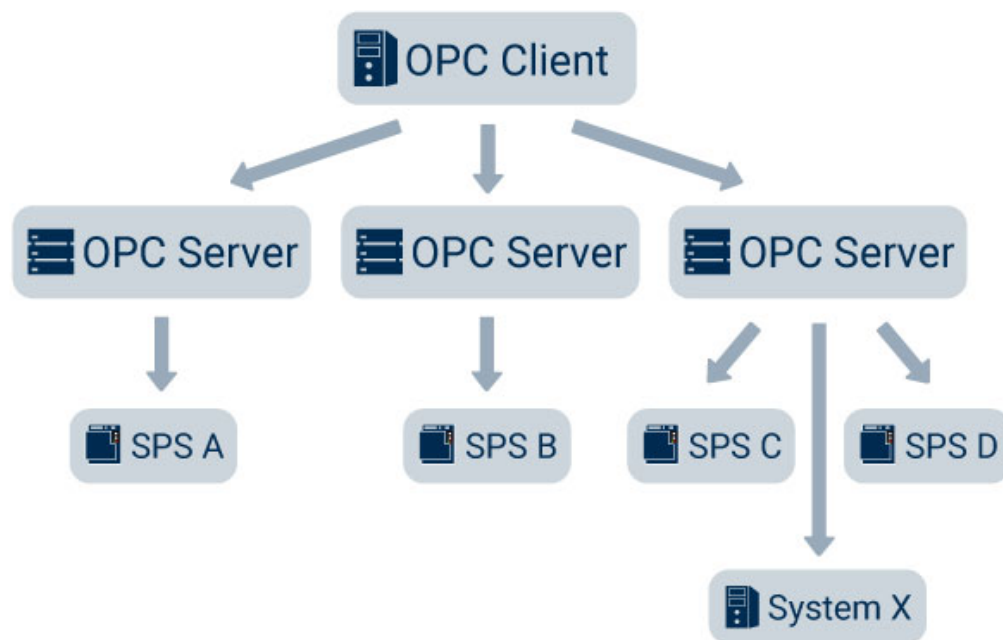
### 2.2 OPC - Unified Architecture

Die in Kapitel 2.1.1 genannten Varianten mit unterschiedlichen Spezifikationen bieten jeweils verschiedene Vor- und Nachteile. Daher entschied sich die OPC Foundation für eine Zusammenführung dieser Spezifikationen zu einem

einheitlichen Standard, wie es das ursprüngliche Ziel vorgab. Anfang 2009 wurde dies als *OPC UA* in die *IEC 62541* aufgenommen. („History OPC“, 10-05-2021) (DIN et al., 17-05-2021a, DIN et al., 17-05-2021b)

### 2.2.1 Aufbau

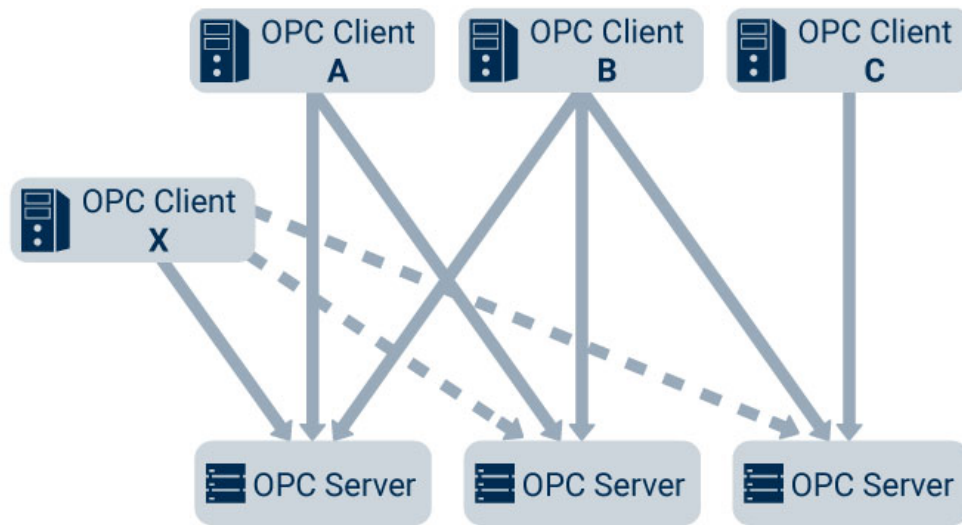
Die Kommunikation durch OPC-UA erfolgt mittels eines Clients, welcher Daten von einem Server abfragt. Dabei wurde zuvor die anzusprechende Anlage serverseitig modelliert. („Möglichkeiten zur Datenmodellierung in OPC UA“, 17-05-2021) Der Serveraufbau erfolgt dabei objektorientiert als Baumstruktur eines voll vernetztes Netzwerk von Knoten. Alle Daten werden vom



**Abbildung 2.1:** OPC Netzstruktur aus Sicht des Servers  
„OPC UA Netzstruktur Serversicht“, 17-05-2021

Server nur bereitgestellt. Der Client muss aktiv auf sie zugreifen. Dadurch kann der Hersteller der Anlage den OPC-Server intern so einrichten, wie es ihm am besten passt, jedoch kann nach außen hin durch herstellerunabhängige Kommunikationsprotokolle ein Datenzugriff erfolgen. Jeder Knoten des Servers kann durch seine ID angesprochen werden. Diese setzt sich zusammen aus einem Namespace-URI, dem Datentyp des Knoten und seinem





**Abbildung 2.2:** OPC Netzstruktur aus Sicht des Clients  
„OPC UA Netzstruktur Clientsicht“, 17-05-2021

Identifizier.

### 2.2.2 Sicherheit

#### 2.2.2.1 Übertragungssicherheit

Die Übertragungssicherheit von Nachrichten und Daten entspricht dem TCP/IP-Standard. („OPC UA und Internet of Things“, 13-05-2021)

#### 2.2.2.2 Zugriffssicherheit

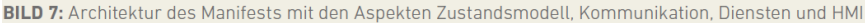
Eine standardisierte und bekannte Netzwerkkonfiguration ist wie im Ethernet-Bereich möglich. („OPC UA und Internet of Things“, 13-05-2021) Die Telegramme können je nach Konfiguration mit 128 Bit oder 256 Bit verschlüsselt werden. Weiterhin sind Benutzer-Authentifizierungen, Signierung von Nachrichten und Paket-Sequenzierungen möglich. Das Bundesamt für Sicherheit in der Informationstechnik prüfte OPC UA 2016 konnte keine systematischen Sicherheitslücken auffinden. („BSI Prüfung OPC UA“, 10-05-2021)

### 2.2.3 Anwendungsbereiche

Durch die standardisierte Abfrage und Senden von Daten an OPC-Server sowie hohe erfüllbare Anforderungen an Zugriffssicherheit ist OPC UA im industriellen Umfeld etabliert und weit verbreitet, unter anderem in der Fördertechnik, der Applikationstechnik, der Prozessindustrie und der Robotik. Eine einfache Konfiguration von Clients auf SCADA-Systemen („SCADA Systeme“, 11-05-2021) und MES-Systemen („MES-Systeme“, 11-05-2021) qualifiziert OPC UA für Anwendungen zum visualisieren und steuern von Prozessen. Hohe Übertragungsraten, wie sie auf Feldebene vorliegen, können derzeit noch nicht erfüllt werden. Eine Alternative zu Feldbussen könnte zukünftig durch EtherCAT und *TSN* entstehen. Perspektivisch besteht ein großes Potential in der Industrie 4.0 durch *OPC UA over TSN*, welches sich derzeit noch in der Entwicklung befindet. Ziel ist es einen einheitlichen Kommunikationsstandard von der Feldebene bis in die Cloud zu schaffen. („OPC UA over TSN“, 13-05-2021) („OPC UA und Internet of Things“, 13-05-2021) („OPC UA und Industrie 4.0“, 13-05-2021)

## 2.3 Module Type Package

Da modulare Anlagen immer wichtiger werden, war es für ein reibungsloses Integrieren notwendig, einen allgemeinen Standard einzuführen. MTP dient dabei als Sprache der Standardisierung um Hersteller- und Technologieneutral modulare Anlagen beschreiben zu können. Es basiert auf AML (Automation Markup Language) einer Weiterentwicklung von UML und wurde in der VDI/VDE/NAMUR 2658 Richtlinien-Reihe spezifiziert. (IEEE, 17-05-2021) Durch eine sogenannte Manifest-Datei werden mittels AML Informationen wie Version, Hersteller, Zustandsmodell, Dienste, Kommunikations- und Benutzerschnittstellen sowie PLT-Stellinformationen beschrieben.



IEEE, 17-05-2021



## 3 Architektur

### 3.1 Komponentendiagramm

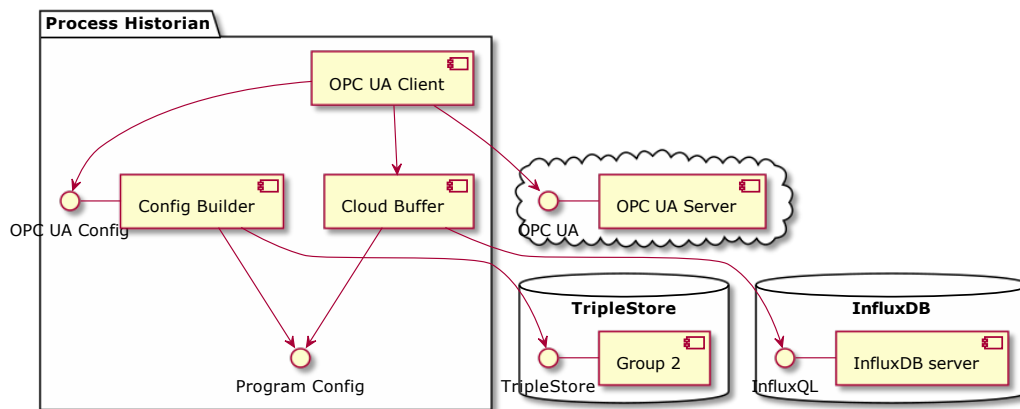


Abbildung 3.1: Komponentendiagramm

Der Process Historian besteht aus drei Komponenten: Dem OPC Client, dem Config Builder und dem Cloud Buffer. Der OPC Client spricht mit der OPC UA-Schnittstelle des OPC UA-Servers. Der Config Builder erstellt aus den Informationen im TripleStore die in Kapitel 3.5 näher beschriebene OPC Config. Der Cloud Buffer speichert die vom OPC Client erhaltenen Daten zwischen und versucht in einem gegebenen Intervall diese in die Influx-Datenbank zu laden. Config Builder und Cloud Buffer greifen weiter auf eine vom Nutzer zu parametrierende Program Config zu. Diese ist ebenfalls in den Konfigurations-Schemata (Kapitel 3.5) näher erläutert. Diese Unterteilung erlaubt eine gute Trennung der Belange und somit eine Wiederverwendbarkeit der Komponenten für ähnliche Systeme.

### 3.2 Klassendiagramm

Im Klassendiagramm spiegelt sich die im Komponentendiagramm (3.1) beschriebene Trennung der Belange wieder. CloudBuffer, OPCUAClient und ConfigBuilder finden sich so nun in eigenen Modulen wieder, die durch

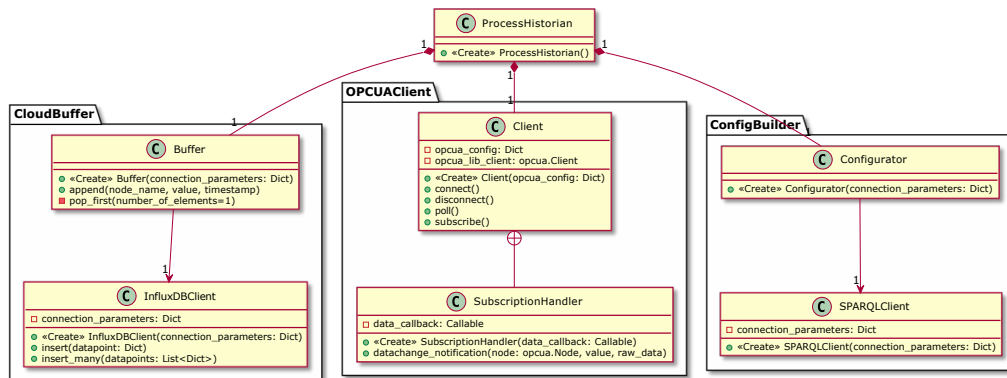


Abbildung 3.2: Klassendiagramm

den ProcessHistorian wiederum zusammengefügt werden. Der CloudBuffer besteht aus einer Buffer-Klasse und einem InfluxDBClient. Der Buffer nimmt werte vom ProcessHistorian entgegen und speichert diese Zwischen. In festgelegten Intervallen versucht er den Pufferinhalt durch den InfluxDBClient in die Datenbank übertragen zu lassen. Ist dies erfolgreich wird er die fehlerfrei gesendeten Daten aus dem Puffer löschen, falls nicht wird er zum nächsten Intervall eine erneute Übertragung des Pufferinhaltes versuchen. Der OPCUAClient besteht aus der Clientklasse selbst, die wiederum eine beinhaltet. Das verhalten des Clients wird durch die vom ConfigBuilder angelegte und in Kapitel 3.5 näher beschriebene Konfigurationsdatei festgelegt. Die SubscriptionHandler-Klasse wird von der OPC UA-Bibliothek benötigt. Der ConfigBuilder besteht aus einem Configurator, der die Konfigurationsdatei für den OPC UA-Client erstellt und einem SPARQLClient, der die Kommunikation mit dem TripleStore übernimmt. Dadurch ist auch innerhalb der Module wieder eine Trennung der Belange gewährleistet. So könnte beispielsweise der InfluxDBClient im CloudBuffer durch einen Client für eine andere Datenbank ersetzt werden.

### 3.3 Sequenzdiagramm

Das Diagramm zeigt den schematischen Ablauf des Programmes. Im Allgemeinen läuft die Erstellung und alle Kommunikation zwischen den Komponenten in der Klasse ProcessHistorian ab. Direkt nach dem Start wird im ersten Schritt der Configurator und damit auch der SPARQLClient aus dem ConfigBuilder-Paket initialisiert. Über diese beiden Komponenten wer-

den die Daten des TripleStores abgerufen und in eine Datei - die OPC UA-Konfiguration - geschrieben. Diese Konfiguration wird im nächsten Schritt bei der Erstellung des Clients des OPCUAClient-Paketes von dieser Klasse direkt geladen. Durch die Initialisierung wird auch die interne Klasse SubscriptionHandler erstellt. Im letzten Schritt der Programminitialisierung wird der sogenannte CloudBuffer zusammen mit seiner InfluxDB Schnittstelle erstellt. Im weiteren Programmverlauf kommt es durch die Möglichkeit Daten vom OPC UA Server zu pollen oder vorher darauf zu subscriben zu unterschiedlichen weiteren Abläufen. In einem Fall werden aktiv vom OPC UA Client nach einer gewissen Zeit alle Daten gepollt, während in dem anderen Fall vom Server eine Benachrichtigung mit den aktualisierten Daten gesendet wird. In beiden Fällen werden die aktualisierten Daten nun über den Process- historian an den CloudBuffer übergeben, der seinerseits nach einer gewissen Zeit alle gesammelten Datenpunkte an die Influx-Datenbank überträgt. Sollte dieser letzte Schritt fehlschlagen, behält er die Daten und versucht es nach einer kurzen Wartezeit erneut.

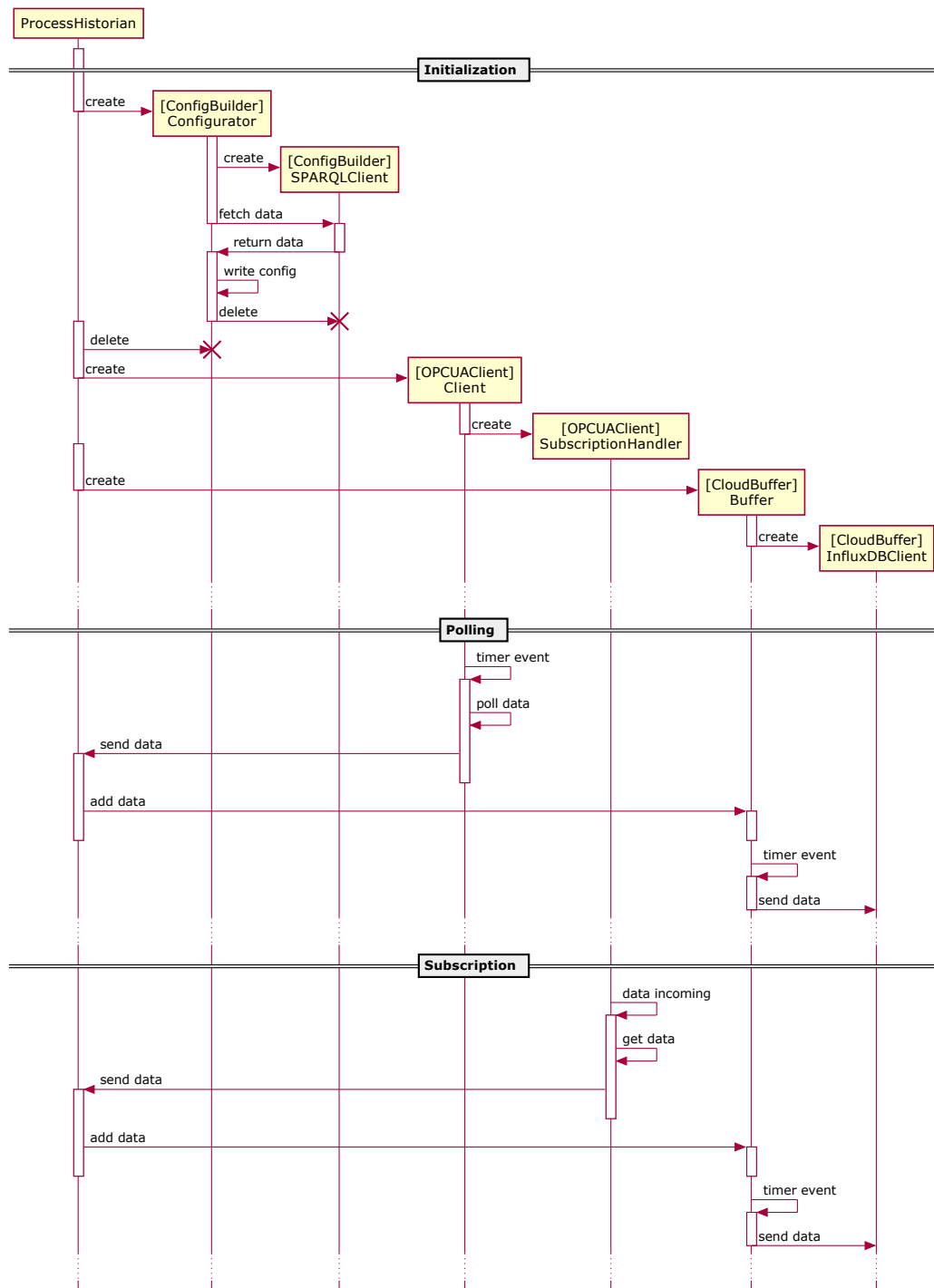
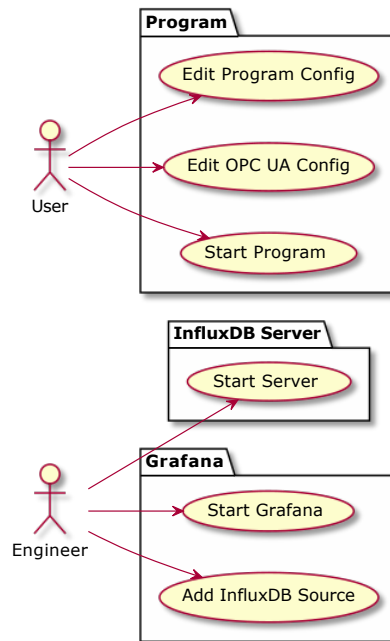


Abbildung 3.3: Sequenzdiagramm



### 3.4 Anwendungsfalldiagramm



**Abbildung 3.4:** Use-Case-Diagramm

Im Anwendungsfalldiagramm ist ersichtlich, dass es zwei getrennte Rollen gibt. Unterschieden wird dabei zwischen dem „User“, der den Process Historian konfigurieren und starten kann und dem „Engineer“, der für die Er- und Bereitstellung der Influx-Datenbank zuständig ist. Während der Entwicklung kann und muss ein Entwickler beide Rollen einnehmen.

### 3.5 Konfigurations-Schemata

Bei den beiden dargestellten Schemata ist die Struktur der Konfigurationsdateien aufgezeigt. In der Programm-Konfiguration werden die notwendigen Parameter eingestellt, die zum Betrieb der Software notwendig sind. Die Einstellungen sind jeweils unterteilt nach ihrem jeweiligen Bereich. Die Namen beschreiben dabei jeweils das Element ausreichend, sodass auch ein unerfahrener Benutzer diese Einstellungen tätigen kann. Die OPC UA-Konfiguration wird automatisch angelegt, wenn die Daten vom TripleStore abgefragt werden, kann aber im Nachhinein von Hand verändert werden, um so z.B. die Abrufmethode für ein Attribut zu ändern.

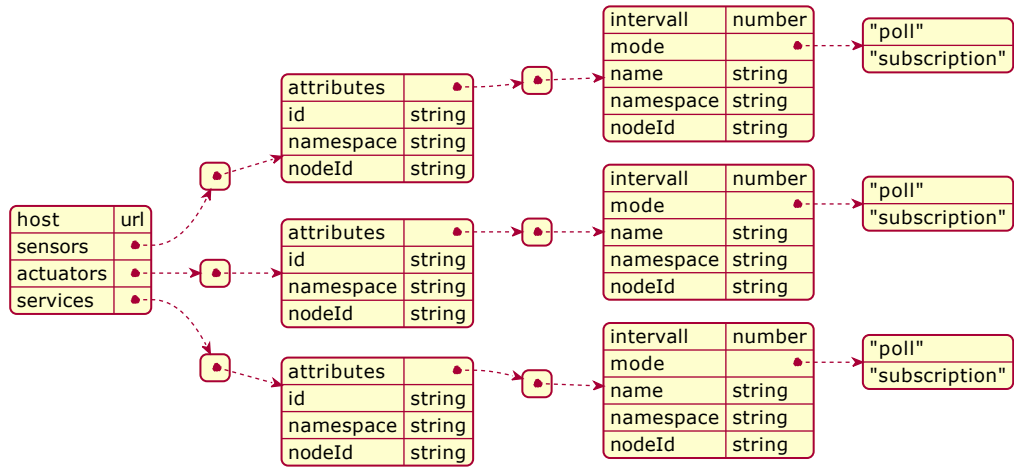


Abbildung 3.5: UPC UA-Config-Schema

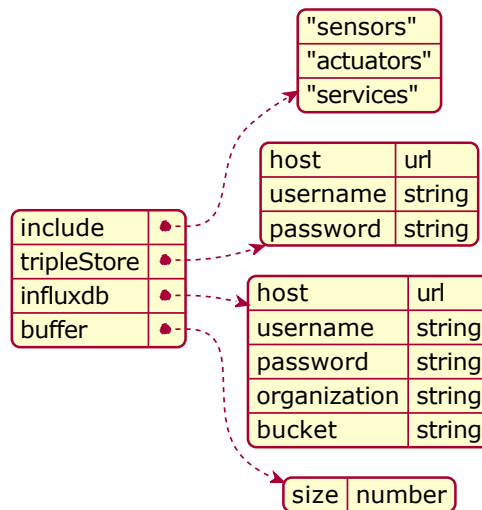


Abbildung 3.6: Program-Config-Schema

## 4 Systemanforderungen

Voraussetzungen für den Betrieb der Software sind:

- Betriebssystem: Linux oder Windows
- Python 3.8+
- CPU: Empfohlen mindestens Dual-Core mit 1 GHz
- Arbeitsspeicher: Experimentell wurden für den CloudBuffer bei einer Größe von 1.000.000 Einträgen eine Speichergröße im RAM von rund 9MB ermittelt.

MacOS als Betriebssystem erfüllt theoretisch alle Anforderungen, kann aber durch uns nicht getestet werden.



## 5 Bedienungsanleitung

Bevor der Benutzer das Programm starten kann, sollte er vorher die Konfigurationsdatei anpassen. Die einzelnen Parameter erklärt:

- `include`: Liste, welche Art von Geräteklassen vom OPC UA Client abgerufen werden soll.
  - Mögliche Werte: „sensors“, „actuators“ und/ oder „services“.
- `tripleStore`: Die Verbindungsparameter zum TripleStore.
- `influxdb`: Die Verbindungsparameter zur InfluxDB.
- `buffer`: Die Größe des CloudBuffers. Alle Einträge nach Erreichen der maximalen Größe verdrängen den ältesten Wert aus der Liste.

Nachfolgend kann der Nutzer das Programm starten mit dem Befehl:

```
$ python3 main.py
```

Alternativ kann die Anwendung auch in einem Docker-Container gebaut und zum Laufen gebracht werden:

```
# docker build -t process_historian
# docker run -v ./config.yaml:/app/config.yaml:r -d process_historian
```

Außerdem existiert eine Docker-Compose-Konfiguration, die die beiden Befehle vereinfacht:

```
# docker-compose build
# docker-compose up -d
```



## Literaturverzeichnis

- BSI Prüfung OPC UA. (10-05-2021). <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/OPCUA/OPCUA.html>. (Siehe S. 9)
- Classic OPC. (10-05-2021). <https://opcfoundation.org/about/opc-technologies/opc-classic/>. (Siehe S. 7)
- DIN, DKE & VDE. (17-05-2021a). DIN EN 62541-4 OPC unified architecture - Part 4: Services (IEC 62541-4:2015). <https://katalog.slub-dresden.de/id/dswarm-201-REUzMDA2NDAwNQ/#detail>. (Siehe S. 8)
- DIN, DKE & VDE. (17-05-2021b). DIN EN 62541-5 OPC unified architecture - Part 5: Information Model (IEC 62541-5:2015). <https://katalog.slub-dresden.de/id/dswarm-201-REUzMDA2NDAwMQ/#detail>. (Siehe S. 8)
- History OPC. (10-05-2021). [https://de.wikipedia.org/wiki/Open\\_Platform\\_Communications](https://de.wikipedia.org/wiki/Open_Platform_Communications). (Siehe S. 7, 8)
- IEEE. (17-05-2021). Namur Modul Type Package Definition. <https://docplayer.org/19029550-Namur-modul-type-package-definition.html>. (Siehe S. 10, 11)
- MES-Systeme. (11-05-2021). <https://www.team-con.de/unsere-leistungen/produkte/manufacturing-execution-system>. (Siehe S. 10)
- Möglichkeiten zur Datenmodellierung in OPC UA. (17-05-2021). <https://www.unified-automation.com/de/news/nachricht-detailansicht/article/extended-opc-ua-modeling-services.html>. (Siehe S. 8)
- OPC Foundation Members. (10-05-2021). <https://opcfoundation.org/members>. (Siehe S. 7)
- OPC UA Netzstruktur Clientsicht. (17-05-2021). [https://www.opc-router.de/wp-content/uploads/2019/07/OPC-Client\\_700x400px.jpg](https://www.opc-router.de/wp-content/uploads/2019/07/OPC-Client_700x400px.jpg). (Siehe S. 9)

- OPC UA Netzstruktur Serversicht. (17-05-2021). [https://www.opc-router.de/wp-content/uploads/2019/07/OPC-Server\\_600x400px.jpg](https://www.opc-router.de/wp-content/uploads/2019/07/OPC-Server_600x400px.jpg). (Siehe S. 8)
- OPC UA over TSN. (13-05-2021). [https://de.wikipedia.org/wiki/OPC-UA\\_TSN](https://de.wikipedia.org/wiki/OPC-UA_TSN). (Siehe S. 10)
- OPC UA und Industrie 4.0. (13-05-2021). <https://opcfoundation.org/wp-content/uploads/2016/05/OPC-UA-Interoperability-For-Industrie4-and-IoT-DE-v5.pdf>. (Siehe S. 10)
- OPC UA und Internet of Things. (13-05-2021). <https://www.industry-of-things.de/was-ist-opc-ua-definition-architektur-und-anwendung-a-727188>. (Siehe S. 9, 10)
- SCADA Systeme. (11-05-2021). <https://www.itwissen.info/SCADA-supervisory-control-and-data-acquisition-SCADA-Protokoll.html>. (Siehe S. 10)
- Web-Tool Timeline der Projektplanung. (04-05-2021). <https://time.graphics/de/line/507245>. (Siehe S. 6)



## Selbstständigkeitserklärung

Hiermit versichern wir, Marc Becker, Erik Noak, Max Kirchner, Patrick Suwinski, Oliver Parczyk, Daniel Kluge, dass wir die vorliegende Projektarbeit im Modul Projektierung von Automatisierungssystemen zum Thema

*Projektierung von Automatisierungssystemen - Entwicklung  
eines OPC-UA basierten Process Historian für modulare  
Anlagen*

ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt haben; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts haben wir Unterstützungsleistungen von folgenden Personen erhalten:

*Dr. rer. nat. Valentin Khaydarov*

Weitere Personen waren an der geistigen Herstellung der vorliegenden Projektierung von Automatisierungssystemen nicht beteiligt.

Dresden, den 17. Mai 2021

.....  
i.A. Patrick Suwinski