VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2022 Jakub Maslowski

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ASSET ADMINISTRATION SHELL PRO PLC

ASSET ADMINISTRATION SHELL FOR PLC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Maslowski
AUTHOR

VEDOUCÍ PRÁCE — Ing. Tomáš Benešl
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Jakub Maslowski ***ID:*** 221003

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Asset Administration Shell pro PLC

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je seznámit se s funkcionalitou a standardy pro AAS. Následně vybrat vhodný komunikační protokol pro komunikaci mezi AAS, navrhnou základní funkcionalitu aktivní části AAS a tu implementovat do PLC.

1. Seznamte se s pojmem AAS, popište jeho typy a funkce.
2. Definujte a popište funkce pasivní části AAS.
3. Definujte a popište základní funkce aktivní části AAS a vytvořte k nim vývojové nebo flow diagramy.
4. Najděte vhodné komunikační protokoly pro komunikaci mezi AAS s přihlédnutím na funkčnost v PLC.
5. Implementujte datový model AAS do PLC.
6. Implementujte základní funkcionalitu aktivní části AAS do PLC.
7. Otestujte své řešení a popište dosažené výsledky.

DOPORUČENÁ LITERATURA:

ZVEI Details of the Asset Administration Shell. [(accessed on 20 August 2021)]; Available online: https://www.zvei.org/en/press-media/publications/details-of-the-asset-administration-shell/

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 23.5.2022

Vedoucí práce: Ing. Tomáš Benešl

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce postupně shrnuje poznatky ohledně I4.0 spolu s jejími základními částmi (AAS, RAMI a DT). Následně se soustředí na popis AAS s ohledem na nezbytné kompromisy od původní myšlenky AAS s cílem jeho zdárné implementace do PLC, které představuje jisté limitace. Práce obsahuje rešerši vhodných komunikačních protokolů pro AAS v PLC.
Praktická část se soustředí na implementaci AAS do PLC od firmy Siemens řady S7-1200 s využitím OPC UA metod. Při tvorbě jsou využity programy SiOME a UaExpert. Práce obsahuje podrobný návod na replikaci AAS implementace spolu s návrhy na možná vylepšení a jejími přínosy v praxi.

KLÍČOVÁ SLOVA

Asset Administration Shell, AAS, Asset, Digitální dvojče, Průmysl 4.0, PLC, OPC UA

ABSTRACT

This bachelor work gradually summarizes the findings regarding I4.0, as well as its basic parts (AAS, RAMI and DT). Subsequently, it focuses on the description of AAS with regard to the necessary compromises from its original idea with the aim of its successful implementation in PLC, which represents some limitations. The work includes solutions of suitable communication protocols for AAS in PLC.
The practical part focuses on the implementation of AAS in PLC from Siemens S7-1200 series using OPC UA methods. SiOME and UaExpert programs are used in the creation. The thesis contains detailed instructions for replication of AAS implementation together with suggestions for possible improvements and its benefits in practise.

KEYWORDS

Asset Administration Shell, AAS, Asset, Digital Twin, Industry 4.0, PLC, OPC UA

Vysázeno pomocí balíčku `thesis` verze 4.03; `http://latex.feec.vutbr.cz`

MASLOWSKI, Jakub. *Asset Administration shell pro PLC*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 51 s. Bakalářská práce. Vedoucí práce: Ing. Tomáš Benešl

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Jakub Maslowski
VUT ID autora:	221003
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Asset Administration shell pro PLC

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Tomáši Benešlovi za odborné vedení, konzultace a podnětné návrhy k práci. Taktéž chci poděkovat své rodině a přítelkyni za jejich podporu.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V posledním desetiletí proběhl velký pokrok v koncepci průmyslu 4.0, který je orientován na služby, což má v případě vhodné implementace následky v podobě zrychlení výroby a větší flexibility systému vůči poruchám. Nová pojetí průmyslu 4.0 jsou aktuálně uplatňována především v Německu, Itálii, Francii, USA, Číně a Japonsku. Nutno podotknout, že čtvrtá průmyslová revoluce se aplikuje do praxe dosti obtížně – jednak zvláště kvůli enormním finančním nákladům na její realizaci, jednak zde hrají podstatnou roli také obavy o bezpečnost.

Jednou z nově vytvořených koncepcí v rámci průmyslu 4.0 je Asset Administration Shell, což si lze představit jakožto digitální dvojče s mnohem větším množstvím komunikačních možností a schopnostmi se samo rozhodovat. Při aplikaci AAS do PLC se mění dosavadní způsob řízení výroby. Řízení na úrovni strojů, kde nadřazené zařízení rozhoduje, co se stane s daným produktem, se přesune do samotných výrobků. Tyto „inteligentní" produkty obsahují veškeré informace o jejich výrobě, nutných úprav a finální podobě.

Téma této práce bylo vybráno z důvodu jejího obrovského potenciálu při uplatnění do praxe. Cílem této práce je navrhnout obecné řešení AAS v PLC, které se dále může replikovat do zařízení v průmyslu.

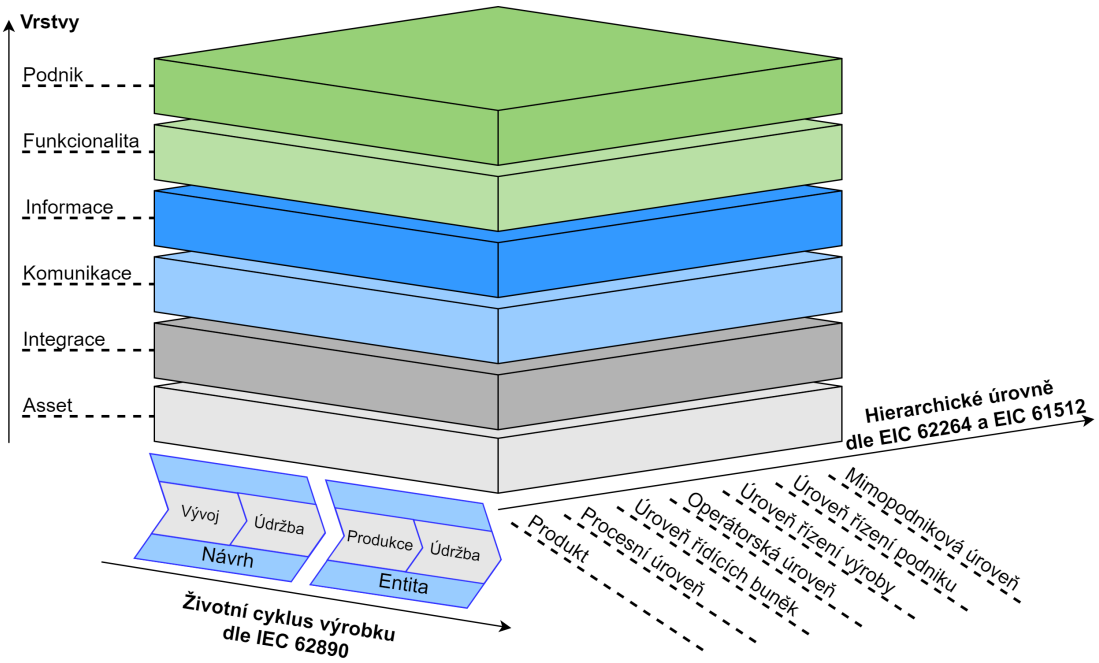
Bakalářská práce se nejprve zabývá nezbytným teoretickým základem nutným pro implementaci AAS do PLC včetně rešerší doporučených standardů, limitacím jejich uplatnění v PLC a popisem chování AAS za pomoci vývojových diagramů. Následně provádí rešerši vhodných komunikačních protokolů pro tuto implementaci. Praktická část obsahuje detailní návod na její replikaci včetně vytvořeného standardu pro tuto problematiku v rámci VUT, způsoby testování a průběžně vysvětluje volby její implementace. Nad rámec zadání obsahuje tato bakalářská práce i souhrn navrhnutých řešení pro části AAS, které aktuálně nejsou předmětem implementace. Práce je členěna do 7 kapitol a obsahuje přílohu s programem v TIA portalu s implementací AAS.

2 Úvod do průmyslu 4.0

Průmysl 4.0 je ovlivněn průběžným vývojem především v oblastech elektrotechniky a informačních technologií. Dosavadní pokrok ve výpočetních technologiích, možnostech uložení velkého množství dat a rozšíření internetu přispělo ke vzniku Průmyslu 4.0. Můžeme definovat šest jeho základních principů, a to interoperabilitu, decentralizaci, virtualizaci, schopnost pracovat v reálném čase (tzn. dodržování Real-Time komunikace), orientaci na služby, modularitu a rekonfigurabilitu.

2.1 Referenční architektonický model pro průmysl 4.0 (RAMI 4.0)

Mezi základní pilíře průmyslu 4.0 patří jeho referenční architektonický model. Tento 3D model, vyvinut z SGAM německými společnostmi ZVEI, VDMA a BITCOM, slouží ke snazší komunikaci a řešení problematik souvisejících s průmyslem 4.0.[19]



Obr. 2.1: Referenční architektonický model pro průmysl 4.0.

Na obrázku 2.1 jsou tři škály jednotlivých postupů, které dohromady tvoří 3D model sloužící jako referenční architektura pro průmysl 4.0. Osa životního cyklu výrobku označuje fáze, ve které se daná entita nachází. Zobrazuje postup produktu od návrhu přes konstrukci, provoz, až k ukončení produkce, případně příslušících

služeb. Následuje hierarchická škála, na které jsou zobrazeny jednotlivé úrovně řízení tak, jak jsou známy z průmyslu 3.0. Na poslední vertikální ose jsou zobrazeny jednotlivé vrstvy, jež jsou nutné k správné funkčnosti průmyslu 4.0. Základ pro komunikaci prostřednictvím AAS uvádí komunikační vrstva, která standardizuje data z integrační vrstvy a umožňuje jejich přenos do vrstev vyšších.

2.2 Digitální dvojče

Názory odborníků na přesnou definici digitálního dvojčete[1] se výrazně rozchází. Jedná se o detailní model fyzického systému snažící se co nejvěrněji simulovat jeho reálné chování v průběhu celého životního cyklu výrobku.[21] Je však nutno podotknout, že samotný 3D model zařízení sloužící k simulaci fyzického systému je technologií průmyslu 3.0, nikoli průmyslu 4.0, přestože je i tento objekt z komerčních důvodů taktéž nazýván jakožto digitální dvojče.[12] Digitální dvojče je možné propojit s jeho fyzickým systémem a umožnit tak vzájemný přenos dat, což přináší hned několik výhod jako například lepší interakci a ukládání dat, ale i porušení kauzality samotného systému, kde za pomoci propojeného DT jsme schopni řídit systém na nasimulovaných datech z budoucna.[17]

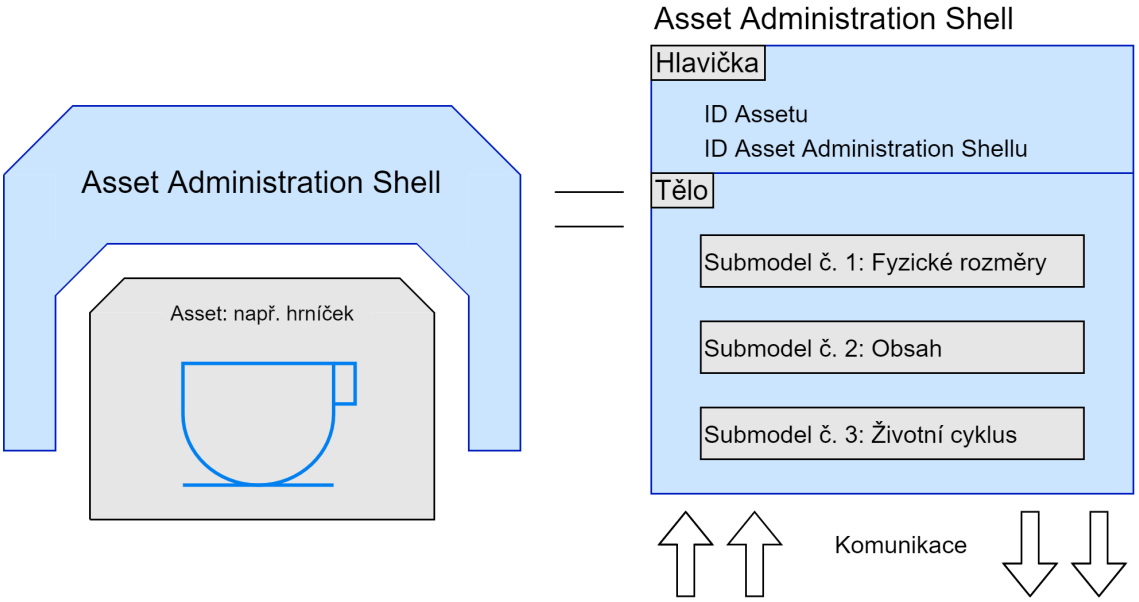
[1] V různých literaturách označováno taktéž jako digitální anděl, digitální stín či virtuální entita.

3 AAS

Asset Administration Shell, též zkráceně označován jako Administration shell [13], nemá v češtině pevně definován překlad. Dal by se volně přeložit jako digitální reprezentace aktiv, administrační panel, správa aktiv a podobně. Všechny tyto překlady jsou dosti nepřesné, což v kombinaci s velmi malým výskytem studií v českém jazyce na toto téma a skutečností, že se v době psaní této práce stále jedná o relativně nový pojem, nastoluje na tuto problematiku lingvistické vakuum.

Asset Administration Shell je v mnoha ohledech podobný digitálnímu dvojčeti. Oproti DT má však AAS daleko větší možnosti komunikace a striktnější pravidla pro uložení jejich dat. AAS taktéž nemusí nutně popisovat dané aktiva po celou dobu jejího životního cyklu, ale pouze v jedné části, což je ostatně případ implementace AAS do PLC.[13] Vzhledem k tomu, že se AAS v určitých aspektech podobá DT a jeho pojmové ukotvení v češtině stále absentuje, může docházet k záměně těchto modelů. Tato skutečnost by pak mohla mít negativní vliv na vzdělávání a přípravu odborníků pro průmysl 4.0 v ČR.

AAS obsahuje veškeré informace o zařízení v průmyslu 4.0 (viz 3.3 Pasivní AAS) a zároveň umožňuje komunikaci s dalšími entitami, které mají rovněž implementovány AAS (viz 3.4 Aktivní AAS). Každá entita v průmyslu 4.0 by měla být reprezentována v AAS pro zajištění bezchybného chodu a dostupnosti informací. Jedná se o základ pro decentralizaci dat a orientaci na služby, což jsou nezbytné kroky pro funkčnost průmyslu 4.0.[5] Jeho schéma je na obrázku 3.1.



Obr. 3.1: Základní schéma AAS.

3.1 Asset

Slovo asset lze do češtiny přeložit jakožto objekt. V tomto případě se jedná o fyzický nebo digitální majetek s určitou hodnotou jako například stroj, licence, dokumentace, ale i lidský operátor.[4] Průmysl 4.0 požaduje platformu, která dokáže takto diverzifikované aktiva uložit na jednotné bázi, což umožňuje právě AAS (viz 3.3 Pasivní AAS).[1]

3.2 ID

Pro vzájemnou komunikaci mezi AAS je doporučeno, aby každý AAS měl své vlastní, unikátní označení. Zde jsou kladeny protichůdné požadavky. Na jednu stranu je preferováno kratší identifikační označení, které by zkrátilo množství dat potřebných k přenosu, což by umožnilo menší zatížení komunikační sítě. Na druhou stranu ovšem samotná koncepce AAS v průmyslu 4.0 uvádí, že by každý asset i jeho Asset Administration Shell měly být identifikovány jednoznačným ID, z čehož v kombinaci s enormním množstvím assetů vyplývá, že tyto identifikační čísla musí být natolik velká, aby nedocházelo k duplicitě.[6]

V rámci AAS jsou k dispozici tři standardy pro identifikaci, a to **IRDI**, definován v ISO 29002-5, ISO/IEC 6523 a ISO/IEC 11179-6, **IRI**, viz [16] a vlastní interní identifikátor o velikosti **UUID**, který má být defikován daným výrobcem.[6] Pro potřeby implementace AAS do PLC by z důvodu ušetření dat požadovaných na identifikaci AAS mohly být využity interní identifikátory kratšího rozsahu jakožto **integer**, případně **long integer**, což by mělo za následek zmenšení datovému toku, a tudíž i rychlejší vzájemné komunikace. Volba jeho formátu musí být vybrána s ohledem na samotný komunikační protokol a způsob, jakým řeší přenos dat.

3.3 Pasivní AAS

Struktura AAS musí splňovat požadavky průmyslu 4.0 a taktéž by měla čerpat z již vyvinutých konceptů.[3] Z tohoto důvodu je k dispozici tabulka 3.1 vyjadřující desatero zásad pro pasívní část AAS, které byly čerpány právě z těchto konceptů. Z tabulky je patrné velké množství pravidel, která nemohou být uplatněna z důvodu limitací PLC.

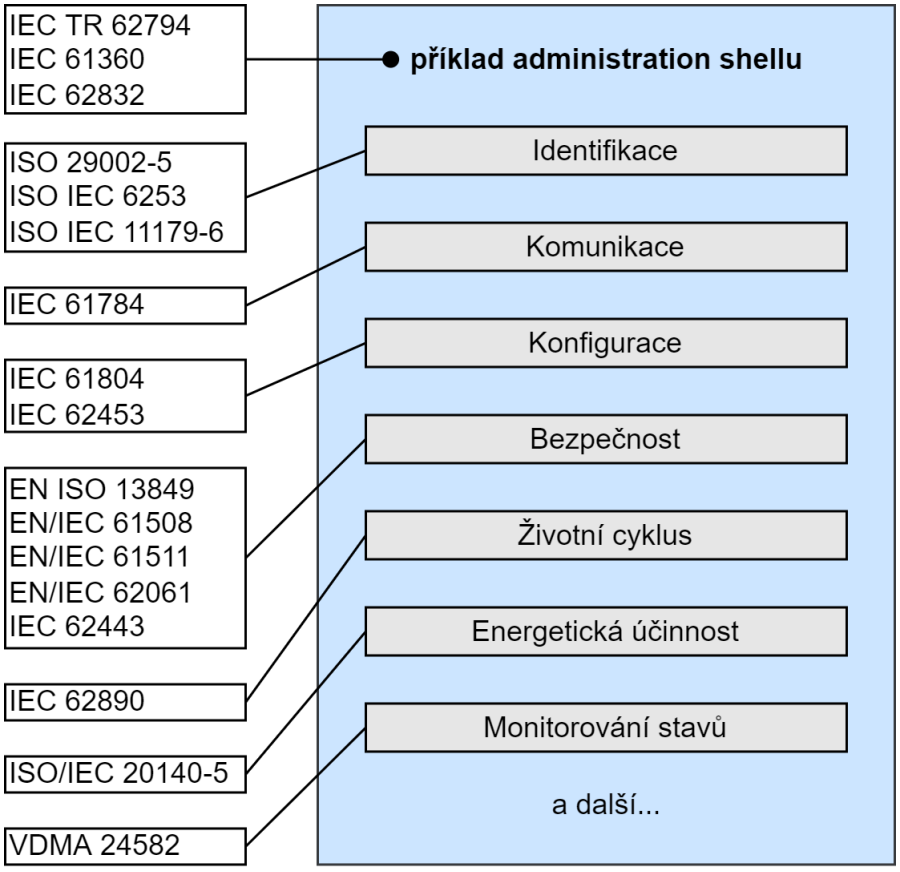
3.3.1 Submodely

Každý submodel AAS musí být standardizován a obsahovat relevantní informace o daném assetu. Jsou využity pro rozdělení informací v digitální podobě do jednot-

Tab. 3.1: Desatero požadavků pasivního AAS.[4][6][14]

ID	Obecné požadavky	Jejich omezení pro implementaci na PLC
1P	Pasivní část AAS se skládá z hlavičky a těla. Hlavička obsahuje informace o identifikaci a tělo informace o assetech v jednotlivých submodelech.	Přenos dat v rámci PLC nelze takto rozdělit.
2P	Struktura AAS je schopna reagovat na požadavky z různých částí průmyslu (obchodní, výrobní, marketingové, aj.).	Struktura AAS v PLC je schopna reagovat pouze na výrobní část průmyslu.
3P	Je nutno využít vhodné implementace pro umožnění přenosu AAS od výrobce k zákazníkovi.	Není součástí implementace AAS do PLC.
4P	Jednotlivé vlastnosti AAS musí být vhodné jak pro návrhy, tak pro samotné entity.	Vlastnosti AAS pro návrhy pouze omezeně.
5P	Mělo by být umožněno vnořování AAS (např. AAS stroje obsahuje AAS jednotlivých součástí).	Není součástí implementace AAS do PLC.
6P	Implementace submodelů by měla umožňovat filtraci zobrazovaných submodelů pro různé pohledy z odlišných částí průmyslu.	Aktuálně nemá PLC software požadované vlastnosti pro dané filtrace.
7P	Jednotlivé submodely mají hierarchickou strukturu.	
8P	AAS povoluje verzování přes jednotlivé submodely.	
9P	AAS musí umožňovat se odkazovat na jiné AAS a své vlastní submodely.	
10P	AAS, Asset i jejich vlastnosti lze jednoznačně identifikovat pomocí omezeného počtu jedinečných identifikátorů (URI, IRDI, UUID).	

livých částí. Submodely se mohou stát standardizovanými, což by umožnilo jejich snadnější použití jakožto vzory. V době psání této práce ovšem tyto standardizace chybí v dostatečné formě, proto v rámci implementace do PLC bude nutné vytvořit takové submodely, které bude následně možno aktualizovat dle budoucích standardů.[8][10] Aktuální standardy, ze kterých je možné vycházet, se nacházejí na obrázku 3.2.



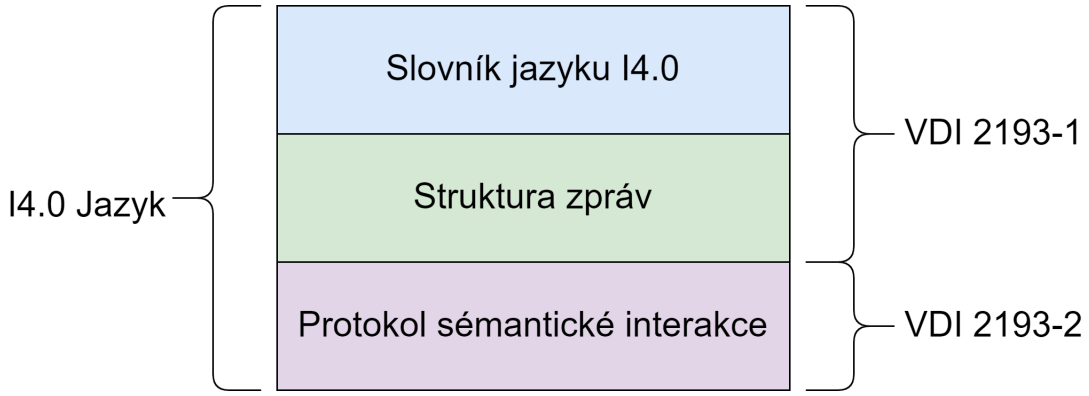
Obr. 3.2: Příklad AAS skládajícího se z submodelů a příslušných standardů.

3.4 Aktivní AAS

Jak již bylo zmíněno, aktivní část AAS se zabývá komunikací, která musí splňovat požadavky na její interoperabilitu, práci v reálném čase a modularitu (tzv. Plug And Produce).[23] Vše je ovšem nutno implementovat v rámci udržení požadované bezpečnosti a zamezení úniku interních informací třetím stranám.

3.4.1 Jazyk I4.0

Jednotné rozhraní pro komunikaci mezi zařízeními definuje VDI 2193-1/2. Její první část definuje tzv. Slovník jazyku I4.0, což má v rámci AAS vitální úlohu, aby nedocházelo ke zbytečnému zdvojení informací v submodelech. Dále pak zmiňuje samostatnou strukturu zpráv pro vzájemnou komunikaci. Část VDI 2193-2 tuto kapitolu dále rozšiřuje o protokol sémantické interakce tak, aby umožňovala implementace I4.0 zařízení do průmyslu.[11] Jeho základní schéma je na obrázku 3.3.



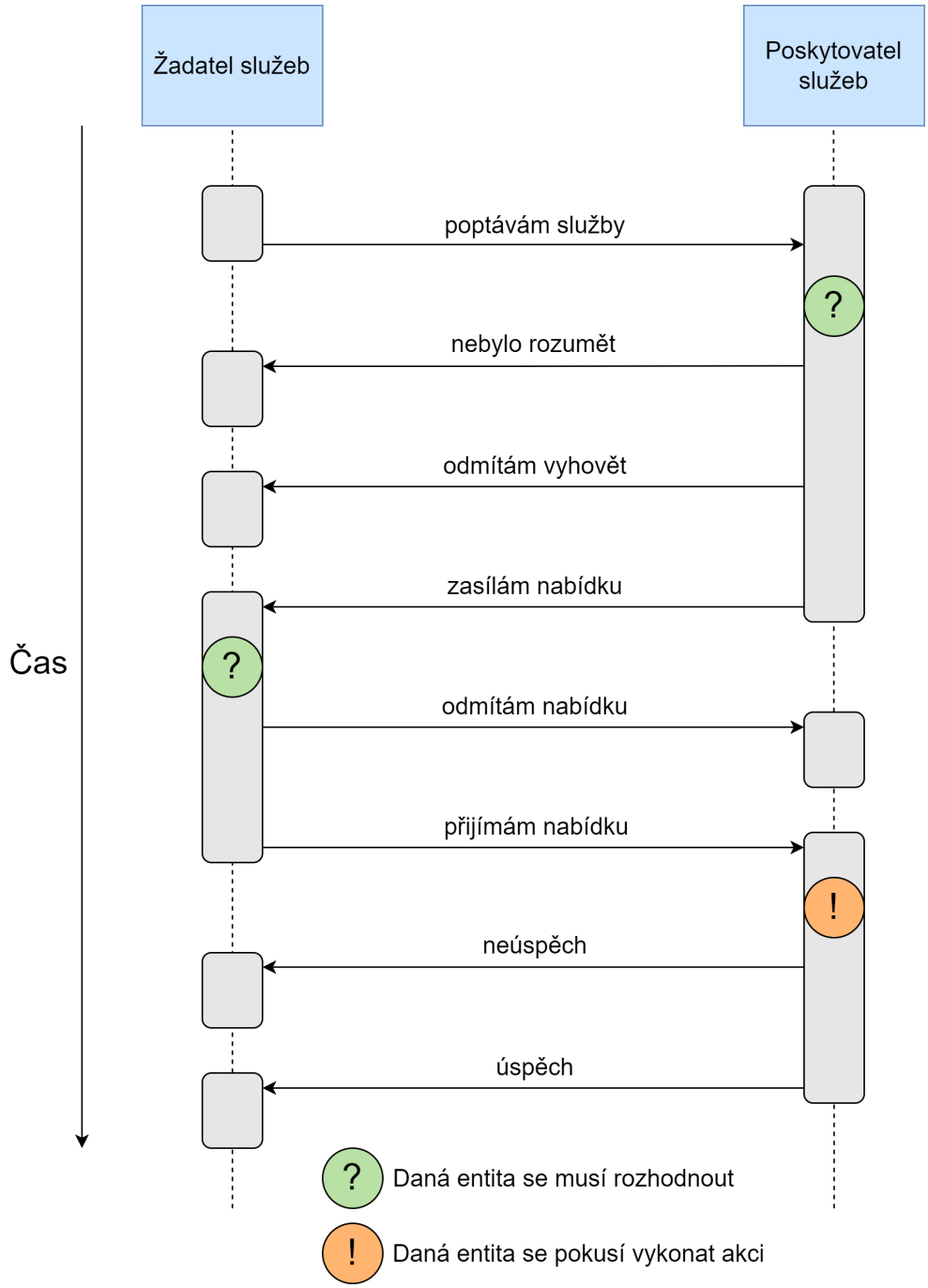
Obr. 3.3: Schéma I4.0 Jazyku.

3.4.2 Komunikace

V rámci komunikace se zařízení dělí na žadatele služeb a poskytovatele služeb. Typický příklad žadatele služeb je výrobek, který žádá po okolních strojích požadované informace, podle kterých se může dle nastavených pravidel rozhodnout kam půjde a co se s ním stane. Poskytovatel služeb naopak může být jakési zařízení, stroj modifikující výrobek, který jim nabízí své služby. Toto dělení na žadatele a poskytovatele však není pevně dané. Poskytovatelé služeb se vždy mohou stát žadateli (kupříkladu stroj žádající obsluhu o opravu), avšak všichni žadatelé se nemohou stát poskytovateli.[9][11] Dle jazyku I4.0 tak, jak jej definoval VDI2193-1/2, probíhá komunikace v tzv. nabídkovém procesu, kde si žadatel a poskytovatel služeb navzájem zasílají předem definované zprávy.[22] Vývojový diagram základní horizontální interakce AAS je na obrázku 3.4, z něhož bude dále vycházeno při návrhu AAS. Ideální vývojové diagramy žadatelů a poskytovatelů služeb jsou na obrázcích 3.5 a 3.6.

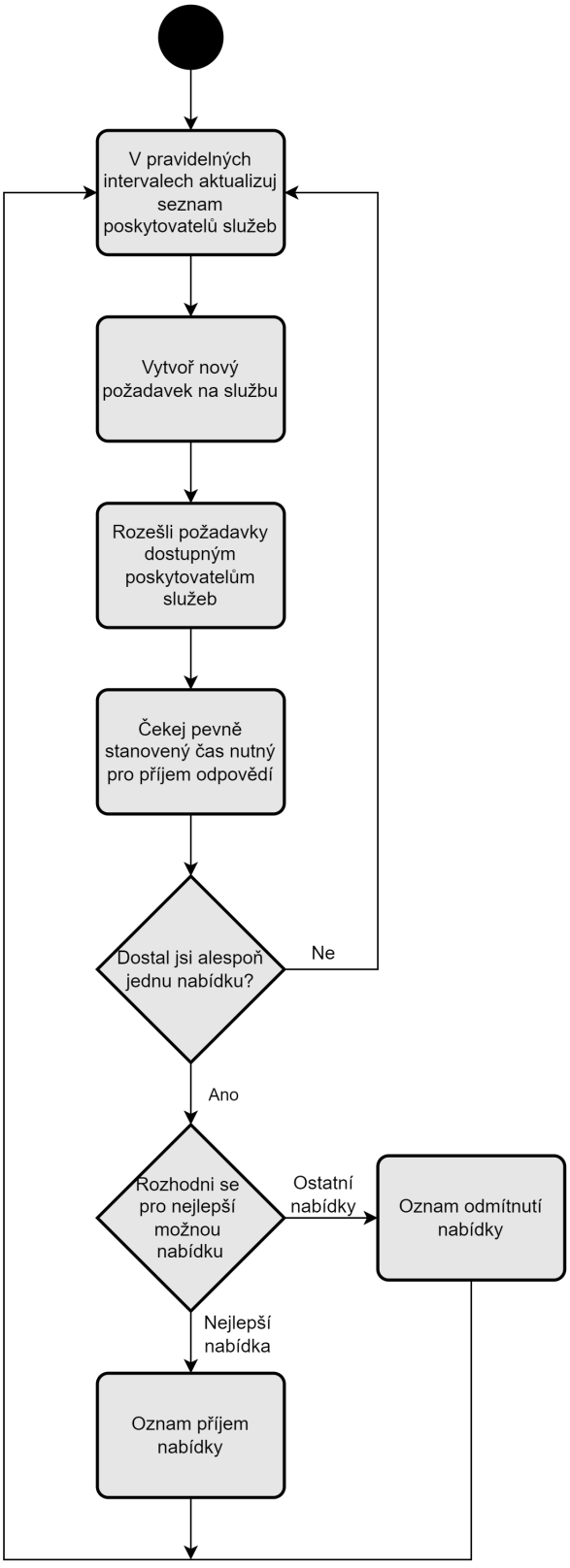
3.4.3 Bezpečnost

Mezi největší zádrhele pro implementace AAS ve výrobních linkách patří bezpochyby obavy o bezpečnost. Jestliže by totiž byla zpřístupněna část AAS v zabezpe-

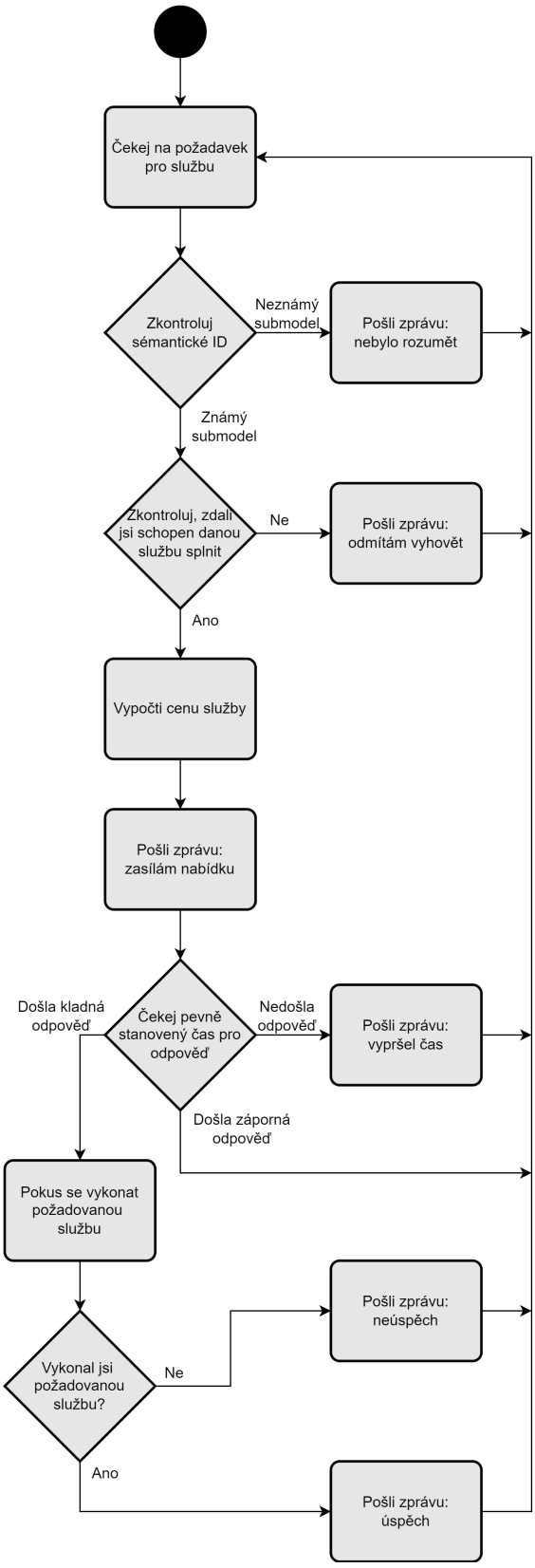


Obr. 3.4: Vývojový diagram základní horizontální interakce AAS.

čené formě mimo interní sítě, tak by mohlo dojít k prolomení daného zabezpečení, zneužití jejich dat či možnosti případných sabotáží. Z tohoto důvodu je v rámci této bakalářské práce omezena veškerá komunikace pouze na výrobky a zařízení za pomoci interních sítí s využitím PLC a volba komunikačního protokolu bude taktéž uskutečněna s přihlédnutím na její bezpečnost.[15]



Obr. 3.5: Vývojový diagram žadatele služeb.



Obr. 3.6: Vývojový diagram poskytovatele služeb.

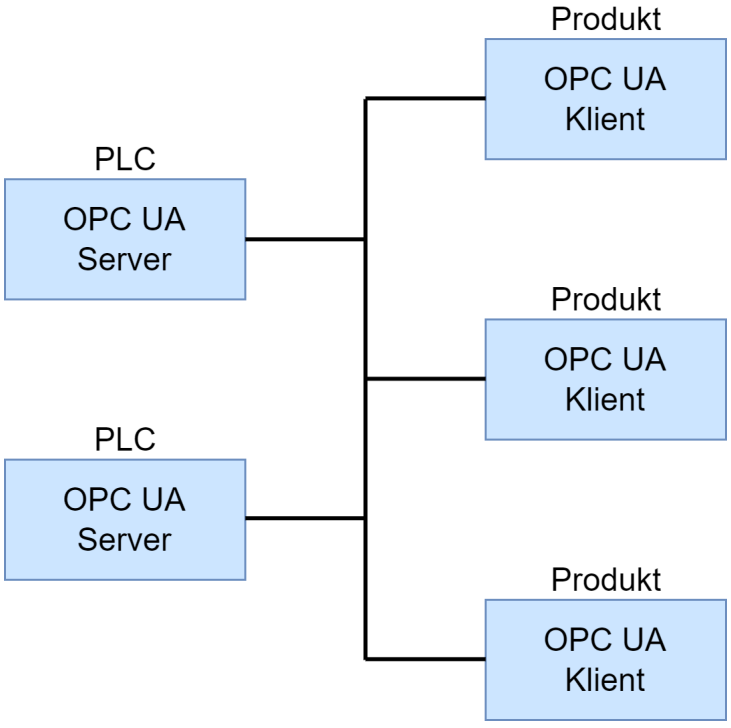
4 Komunikační protokoly

Pro implementace vzájemné komunikace AAS s PLC jsou k dispozici dvě výrazné platformy. Následující kapitoly se zabývají jejími výraznými rysy.

4.1 OPC UA

OPC Unified Architecture je platformě nezávislá architektura působící napříč ISO OSI vrstvami komunikace. Jedná se o komunikační platformu zaměřenou na služby, která z podstatné části splňuje již zmíněné požadavky na průmysl 4.0. Každé zařízení se chová jako server nebo klient. Schéma jejich vzájemné komunikace je na obrázku 4.1. Klient se připojuje na požadovaný server, kde může proběhnout přenos dat.[1]

Pro AAS se jeví jako vhodná její modifikace OPC UA: LDS-ME (Local Discovery Server with Multicast Extensions), kde každé zařízení obsahuje vlastní server a zároveň je díky vzájemnému ohlašování se umožněno Plug and Produce. PLC však tuto modifikaci aktuálně nepodporují a v případě zvolení OPC UA by bylo nutné využít základní variantu.[5][18][20]



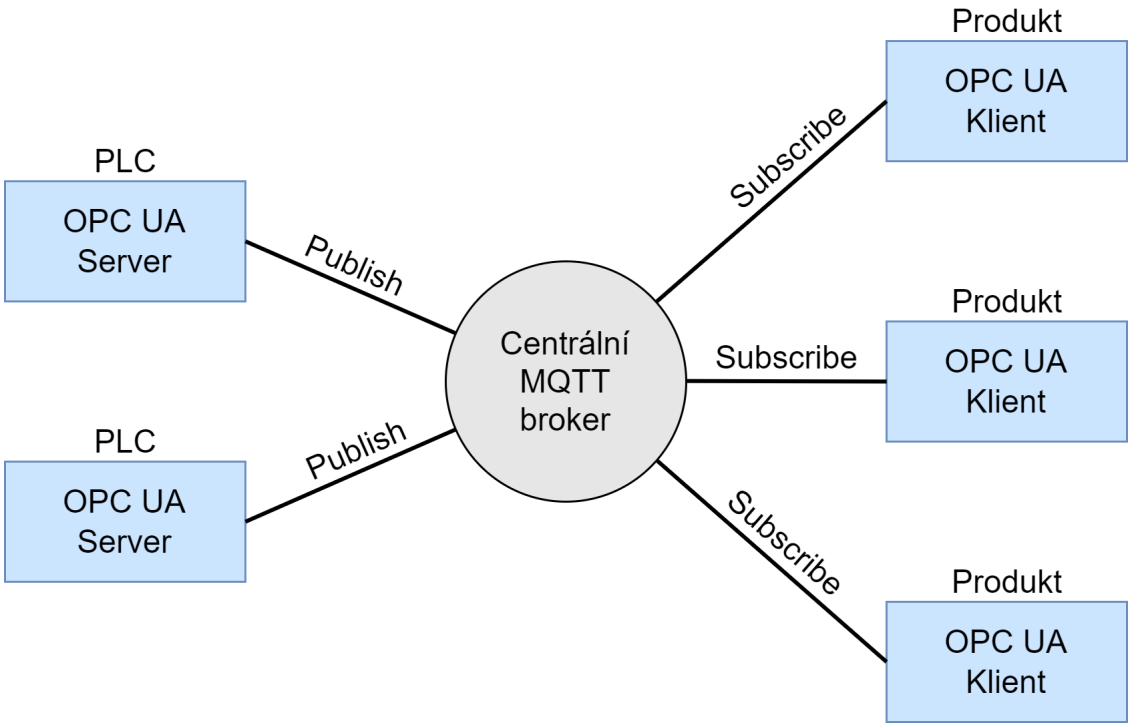
Obr. 4.1: Komunikace v rámci protokolu OPC UA.

[1]Od verze specifikace 1.04 je taktéž možné využít infrastrukturu typu Publisher-Subscriber.

4.2 MQTT

Message Quering Telemetry Transport je síťový protokol založený na principu Publisher - Subscriber. Veškerou komunikaci přes tuto síť zajišťuje centrální MQTT broker, který filtruje proudící zprávy a dále zařízením přeposílá zprávy pro ně určené, jež se nacházejí v žádaných tématech.

Tento způsob komunikace je vhodný především pro IoT zařízení. Naopak pro AAS nemusí být výše popsaná komunikace vhodná, neboť centrální broker může způsobovat opožďování zpráv při větším množství zařízení, což by mělo výrazně negativní dopad na celkovou funkčnost AAS.[5][7][11]



Obr. 4.2: Komunikace v rámci protokolu MQTT.

4.3 Volba komunikačního protokolu pro AAS v PLC

OPC UA má velkou výhodu díky objektové reprezentaci dat. Klienti můžou dokonce volat jeho metody. Další jeho předností je umožnění Real-Time komunikace, což je pro ovládání PLC taktéž zásadní.

MQTT má výhody v podobě její malé velikosti, což má pozitivní efekt na nutnou velikost komunikační sběrnice, a nízkých požadavků na výpočetní výkon zařízení. Tyto výhody se v PLC moc neprojeví. Jeho velká nevýhoda je centralizovaný prvek

v podobě MQTT brokeru, který především při větším provozu znemožňuje Real-Time komunikaci.

Oba komunikační protokoly jsou podporovány v PLC od firmy Siemens. Vzhledem ke skutečnosti, že výhody protokolu OPC UA dalece převyšují benefity užití MQTT v PLC, vybral jsem pro implementaci AAS do PLC komunikační protokol OPC UA.

5 Praktická část / Implementace

Kvůli omezením PLC je na jednom CPU implementováno pouze jedno AAS, a tedy může zaštiťovat taktéž pouze jeden Asset. Z tohoto důvodu bylo v implementaci upuštěno od rozlišování ID samotných assetů a AAS. V případě potřeby lze jednoduše přidat další ID do pasivní části AAS, avšak nepřineslo by to žádné významné benefity.

5.1 PLC

V rámci implementace AAS do PLC byly vybrány PLC od firmy Siemens z důvodu jejich rozšířenosti a široké podpoře. V průběhu tvorby AAS byly využity PLC řady S7 - 1500, neboť pro ně existovala možnost simulace na PLCSIM Advanced V4.0, což výrazně zjednodušovalo práci oproti využití fyzických PLC. V závěru tvorby AAS byl tento kód přenesen a optimalizován pro reálné PLC řady S7 - 1200, aby ho následně bylo možné jednoduše napojit na testbed Barman, nacházejícím se na ústavu automatizace a měřící techniky VUT v Brně.

Nutno podotknout, že pro zmíněnou řadu lze kód přenést pouze do PLC podporující OPC UA s firmwarem verze 4.45 a vyšší. (Jedná se o PLC typu 6ES7214-1xx40-0XB0.) Starší verze firmwaru totiž plně nepodporují OPC UA metody, které jsou v rámci implementace AAS naprosto nezbytné. Jestliže by bylo nutné kód zreplikovat pro PLC řady S7 - 1500, tak je nezbytný firmware verze 2.5 a vyšší. Celý kód byl vytvořen v TIA portalu V17, metody jsou však podporovány již od verze V15.1.

5.2 Rozdělení poskytovatelů služeb (SP)

Z rozdělení AAS na SR a SP v kombinaci s nemožností vytvoření univerzálního AAS, které by umělo obojí, vyplývá nutnost tvorby programu pro každou skupinu zvlášť. Z důvodu limitací PLC byly implementovány pouze poskytovatelé služeb reprezentující stroje v továrně. Žadatelé služeb, reprezentující výrobky v továrně, které by volaly metody poskytovatelů služeb, nejsou součástí této práce, neboť by pro ně byla vhodnější jiná platforma, nežli již zmiňované PLC.

Poskytovatelé služeb byli rozděleni na 3 podskupiny, do kterých je možno zařadit jakékoliv zařízení v továrně. Jsou jimy:

Storage: Jedná se o skladovací jednotku, která může vydávat materiály jak do výroby, tak na export. Typicky stojí u vzniku AAS.

Manipulator: Jedná se o zařízení provádějící transport výrobků.

Manufacturing_unit: Pod tuto podskupinu lze zařadit výrobní jednotku, která provádí změny na výrobcích.

V programu jsem vytvořil pro každý typ AAS samostatný program pro dané PLC. Moje AAS implementace tedy nepodporuje jedno zařízení chovající se jako 2 a více typů poskytovatelů služeb zároveň. Kupříkladu jedno PLC nemůže ovládat jak manipulátor, tak samotnou výrobní jednotku. Jestliže by tato situace nastala v praxi, je možné přiložený kód jednoduše upravit tak, aby zařízení mohlo poskytovat služby různých podskupin poskytovatelů služeb. Toto řešení nebylo v rámci mé implementace využito z důvodu, že výsledný kód by byl méně efektivní a náročnější na pochopení, což by se příčilo se samotným cílem této bakalářské práce, která má navrhnout, otestovat a implementovat vzorové řešení AAS v PLC, které následně po individuálních úpravách dle specifických požadavků dané implementace může být využito v průmyslu.

5.3 Univerzální rozdělení AAS metod

Každé AAS je složeno z 5 metod zajišťujících aktivní část AAS a několika sdílených, aktuálně statických dat. V rámci snahy o univerzálnost a jednotu má každá OPC UA metoda maximálně jeden vstupní a jeden výstupní string[1], v němž jsou pomocí znaku / odděleny jednotlivé argumenty daných funkcí. Číselné kódování ID jednotlivých pozic, materiálů a případných operací je v tabulce 5.1.

- Metoda **GetSubmodel** nemá žádný vstupní parametr a vrací string o možných hodnotách „Storage", „Manipulator" či „Manufacturing_unit" v závislosti na tom, o jaké AAS se jedná. Tato metoda se dá nahradit nasdílením proměnné **Type** do OPC UA prostoru spolu s pasivní částí AAS. Tohoto nebylo využito pro názorné předvedení a popis funkce v programu pro případ metody bez vstupních parametrů.
- Metoda **Reserve** má jeden vstupní parametr **SR ID**, což je string reprezentující identifikaci žadatele služeb. Tento string nemusí být nutně pouze číselný s ohledem na doporučený formát z ZVEI, avšak pro praktické použití a přehlednost se doporučují používat znaky pouze v minimální míře. V rámci mé implementace v něm nesmí být znaky /, neboť se jedná o rozdělovací znak a při volání této metody předpokládám, že může nastat situace, kdy ze strany žadatele služeb přijde vstupní string ve tvaru **SR ID/další slova**. Metoda zaštiťuje rezervaci zařízení pro využití služeb daného SP. Aktuálně nejsou implementovány žádné rezervační fronty, což při větším množství žadatelů slu-

[1]Jedná se o parametry OPC UA metody, nikoliv o parametry funkčních bloků! Funkční bloky mají přístup k těmto parametrům přes specifické knihovní funkce. Jejich vstupní/výstupní parametry jsou dány potřebou interní komunikace s kódem starajícím se o chod stroje.

Tab. 5.1: Popis číselně kódovaných ID proměnných v programu.

ID materiálu	Význam
1	Malá sklenička
2	Velká sklenička
3	Voda
4	Rum
5	Led
6	Tequila
7	Džus
8	Citronová šťáva
9	Zamíchání
ID pozice	Význam
0	Chybná pozice
1	Výdejní pozice skladovací jednotky
2	Pozice sféry vlivu shakeru
3	Pozice sféry vlivu dávkovače
4	Pozice sféry vlivu jednotky na výdej ledu
5-10	Výdejní pozice nápojů

žeb může vést k neefektivnímu plánování výroby. Zařízení se samo odrezervuje od daného žadatele, jestliže je rezervováno a zároveň nevyužito déle než 2 minuty bez přerušení. Toto je pro případ, kdy by si žadatel služeb zarezervoval dané zařízení a následně by došlo k jeho destrukci či nekorektnímu odrezervování. V takovém případě by se zařízení dostalo do dead locku a nebylo by možné s ním nadále pracovat. Tato metoda vrací string o hodnotách „TRUE" či „FALSE" v závislosti na tom, zdali se dané AAS úspěšně zaregistrovalo.

- Metoda **Free** má totožnou strukturu vstupních i výstupních parametrů jako metoda Reserve, včetně stejné pojistky pro případ, kdy ve vstupní zprávě bude více slov, nežli pouze **SR ID**. Jestliže je zaregistrován stejný žadatel jako ten ve vstupní zprávě, nebo není zaregistrován nikdo, tak vrací zprávu „TRUE", v opačném případě pak „FALSE".
- Metoda **Give** je dostupná pouze pro SP typu **Storage**. Vstupní string má strukturu **SR ID/číselné ID materiálu/číselné množství** a vrací zprávu „TRUE" či „FALSE" dle zahájení dané operace. Popis jejich vstupních a výstupních parametrů je v tabulce 5.2. Průběžný postup zapisuje do proměnné **GiveStatus** typu string, která může nabývat hodnot

  - „**idle**” - zařízení je v klidu
  - „**inProgress**” - zařízení vykonává určitou činnost
  - „**done**” - zařízení úspěšně vykonalo požadovanou činnost, stále je zarezervováno žadatelem služeb pro jeho další využití
  - „**failed**” - zařízení neúspěšně vykonalo požadovanou činnost, stále je zarezervováno žadatelem služeb pro jeho další využití

Tab. 5.2: Popis vstupních a výstupních parametrů funkčního bloku Give napojených na Device.

Typ parametru	Název	Datový typ	Popis
Input	DeviceDone	Bool	Jeho hodnota se očekává true, jestliže zařízení ukončilo svou dosavadní činnost a jeho GiveStatus je v hodnotách „done” / „failed” / „idle”.
Output	Material	Int	Obsahuje číselné označení materiálu, který má sklad vydat, viz tabulka 5.1.
Output	Amount	Int	Obsahuje množství materiálu, který má sklad vydat.
Output	Valid	Bool	Proměnná říkající skladovací jednotce, že dané parametry jsou validní a může dle nich provádět požadovanou činnost.

- Metoda **Transport** je dostupná pouze pro SP typu **Manipulator**. Vstupní string má strukturu **SR ID/číselné označení materiálu/číselná počáteční pozice/číselná konečná pozice** a vrací zprávu „TRUE” či „FALSE” dle zahájení dané operace. Popis jejich vstupních a výstupních parametrů je v tabulce 5.3. Průběžný postup zapisuje do proměnné **TransportStatus**, která může nabývat totožných hodnot jako proměnná **GiveStatus**.
- Metoda **Change** je dostupná pouze pro SP typu **Manufacturing_unit**. Vstupní string má strukturu **SR ID/číselné ID materiálu/číselné množství** a vrací zprávu „TRUE” či „FALSE” dle zahájení dané operace. Popis jejich vstupních a výstupních parametrů je v tabulce 5.4. Průběžný postup zapisuje do proměnné **ChangeStatus**, která může nabývat totožných hodnot jako proměnná **GiveStatus**.
- Metoda **GetSupported** slouží k evaluaci nabízených služeb. Struktura vstupních parametrů je totožná jako struktura jejich přidružených specifických metod **Give**, **Transport** či **Change**. Například pro SP typu **Manipulator** má

Tab. 5.3: Popis vstupních a výstupních parametrů funkčního bloku Transport napojených na Device.

Typ parametru	Název	Datový typ	Popis
Input	DeviceDone	Bool	Jeho hodnota se očekává true, jestliže zařízení ukončilo svou dosavadní činnost a jeho TransportStatus je v hodnotách „done" / „failed" / „idle".
Output	Material	Int	Obsahuje ID materiálu, který má manipulátor přenést.
Output	InitPosition	Int	Obsahuje ID pozice, ze které má manipulátor daný výrobek přenést.
Output	FinalPosition	Int	Obsahuje ID pozice, do které má manipulátor daný výrobek přenést.
Output	Valid	Bool	Proměnná říkající manipulátoru, že dané parametry jsou validní a může dle nich provádět požadovanou činnost.

tato metoda vstupní string ve tvaru **SR ID/číselné označení materiálu-/číselná počáteční pozice/číselná konečná pozice**. Výstup této metody je bezrozměrné číslo v rozmezí **0 - 100**. Čím větší je číslo, tím více podporuje dané zařízení požadovanou službu. Jestliže vrátí metoda hodnotu 0, tak tuto službu daná metoda aktuálně nepodporuje. Takto si může výrobek dle jejich individuálních nabídek vybrat optimálního SP, u kterého si danou službu objedná.

5.4 Struktura programu

V následujících podkapitolách jsou popsány jednotlivé části programu, které jsou nezbytné pro správnou funkčnost AAS v PLC.

5.4.1 Metody

Tvorba metod vycházela z dostupného návodu pro PLC řady S7 - 1500 [2]. OPC UA metody se programují jako klasické funkční bloky (FB), do kterých se navíc přidají knihovní funkce **OPC_UA_ServerMethodPre** a **OPC_UA_ServerMethodPost**, které jsou dostupné pod záložkou **Communication/OPC UA/OPC UA**

Tab. 5.4: Popis vstupních a výstupních parametrů funkčního bloku Change napojených na Device.

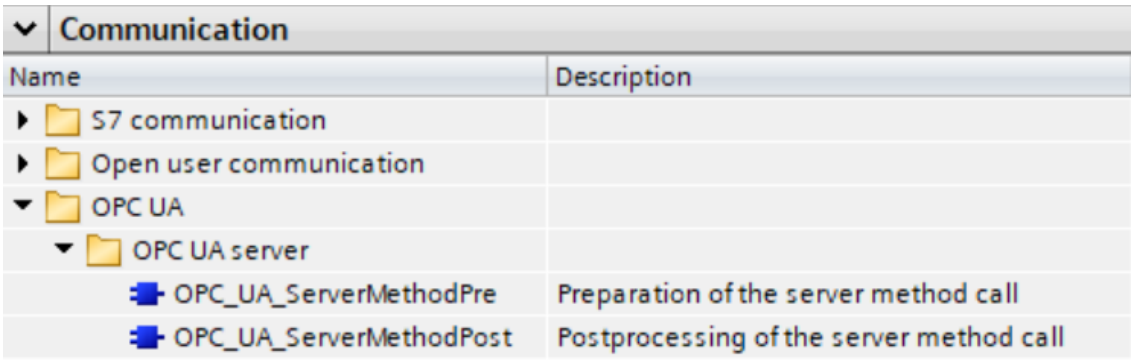
Typ parametru	Název	Datový typ	Popis
Input	DeviceDone	Bool	Jeho hodnota se očekává true, jestliže zařízení ukončilo svou dosavadní činnost a jeho ChangeStatus je v hodnotách „done” / „failed” / „idle”.
Output	Material	Int	Obsahuje ID materiálu či operace, která se má vykonat. Vzhledem k uplatnění na testbedu Barman a limitacím PLC jsou definovány pouze materiály a operace pro něj podstatné (viz tabulka 5.1).
Output	Amount	Int	Jedná se o množství, které více specifikuje požadovaný materiál/operaci. Jednotky jsou pro potřeby dávkovače, který je definován v programu, zpracovávány ve formě mililitrů. Pro implementaci tohoto typu poskytovatele služeb na shaker se doporučuje použití jednotek milisekund.
Output	Valid	Bool	Proměnná říkající výrobní jednotce, že dané parametry jsou validní a může dle nich provádět požadovanou činnost.

Server, viz obrázek 5.1.

Tyto knihovní funkce zajišťují, že se daný funkční blok chová jako OPC UA metoda a je možno ho následně namapovat do OPC UA prostředí. Starají se tedy o vstupní a výstupní parametry metod přes pevně pojmenované proměnné v tabulce 5.8.

Při volání metody se vstupní parametry přesunou do kníhovní funkce **OPC_UA_ServerMethodPre**, následně se vykoná samotný kód metody a výsledky se zapíší do funkce **OPC_UA_ServerMethodPost**, která uvědomí OPC UA klienta. Vstupní a výstupní parametry metod jsou limitované podporovanými datovými typy, uvedenými v tabulce 5.5.

V tabulkách 5.6 a 5.7 jsou definovány paramtery použitých knihovních funkcí pro tvorbu OPC UA metod. Paremetry **Status** jsou nastavovány dle **OPC UA Tag**



Obr. 5.1: Zobrazení knihovních funkcí pro OPC UA metody v TIA portalu.

Tab. 5.5: Přehled povolených davotých typů pro **UAMethod_InParameters** a **UAMethod_OutParameters** knihovních funkcí pro OPC UA.

Datový typ	Reprezentace v TIA portalu
String	WSTRING
Boolean	BOOL
SByte	SINT
Int16	INT
Int32	DINT
Int64	LINT
Byte	USINT
UInt32	UDINT
Uint64	ULINT
Float	REAL
Double	LREAL
DateTime	LDT (Nedostupné pro S7 - 1200)

Table, viz podsekce 5.4.5 PLC tagy.

Každá OPC UA metoda musí mít přesně definované proměnné z tabulky 5.8, kde je mimo jiné i nutnost uložení předepsaných knihovních funkcí jako multi-instance pro správnou funkčnost OPC UA metod v programu. Nad rámec těchto proměnných se mohou v metodách objevit statické a dočasné proměnné jako jsou časovače, jemu příslušné proměnné apod., které zajišťují individuální požadavky pro různé metody.

Na začátku programu metody je nutné zavolat knihovní funkci **OPC_UA_ServerMethodPre_Instance** a do jejího vstupního parametru **UAMethod_InParameters** přiřadit proměnnou **UAMethod_InParameters**, která je defino-

Tab. 5.6: Popis rozhraní knihovní funkce **OPC_UA_ServerMethodPre**.

Typ parametru	Název	Datový typ	Popis
Output	Done	Bool	Vrací true při úspěšném zavolání funkce.
Output	Busy	Bool	Vrací true při zpracování funkce.
Output	Error	Bool	Vrací true, jestliže se naskytne chyba při volání funkce.
Output	Status	DWord	Popis chyby, která nastala.
Output	UAMethod_Called	Bool	Vrací true, jestliže byla metoda zavolána klientem.
InOut	UAMethod_InParameters	Variant	V našem případě se jedná vždy o string zadaný klientem, detailnější popis možných hodnot viz tabulka 5.5.

vána jako samostatná datová struktura. Její výstupní parametry je taktéž nutné uložit do proměnné **statUAM_Status_Pre**.

Samotný kód metody je uvnitř IF podmínky, která kontroluje, zdali **UAMethod_Called** je roven true. Uvnitř je vždy zavolána interní funkce **GetMessage**, která vstupní zprávu od klienta rozdělí až na pět samostatných slov, které byly v původní zprávě odděleny znakem /. Tato zpráva je uložena do uživatelského datového typu **typeArrayMessage** složeného ze struktury obsahující 5 stringů. Mimo tuto IF podmínku je vhodné definovat aspekty programu, které se vykonávají bez ohledu jejího volání typicky časovače.

Na konci programu metody je nutné zavolat knihovní funkci **OPC_UA_ServerMethodPost_Instance** a do jejich vstupních parametrů přiřadit odpovídající proměnné z **tempUAM_MethodHandling** a **UAMethod_OutParameters**. Její výstupní parametry je nutné uložit do proměnné **statUAM_Status_Post**.

5.4.2 Volání aktivního AAS v programu

Funkční blok **Active AAS** zaštiťující veškeré OPC UA metody potřebné pro splnění požadavků OPC UA klienta je složen z cyklycky volajících se funkčních bloků, které obsahují dané OPC UA metody, díky zmíněným knihovním funkcím. Jeho vstupní

Tab. 5.7: Popis rozhraní knihovní funkce **OPC_UA_ServerMethodPost**.

Typ parametru	Název	Datový typ	Popis
Output	Done	Bool	Vrací true při úspěšném zavolání funkce.
Output	Busy	Bool	Vrací true při zpracování funkce.
Output	Error	Bool	Vrací true, jestliže se naskytne chyba při volání funkce.
Output	Status	DWord	Popis chyby, která nastala.
Input	UAMethod_Result	DWord	Jedná se o kód OPC_UA statusu, který je definován v **OPC UA Tag table**.
Input	UAMethod_Finished	Bool	Metoda ukončí svou činnost, jestliže se nastaví na true.
InOut	UAMethod_OutParameters	Struktura	V našem případě se jedná vždy o string poslaný klientovi. Detailnější popis možných hodnot viz tabulka 5.5.

a výstupní parametry jsou ze specifických metod **Transport**, **Change** či **Give**, které musí být v aplikaci propojeny s kódem starajícím se o chod stroje, což je v mé aplikaci provedeno přes datový blok **ProgramData**. Na ně jsou požadavky, aby měnil proměnnou **xxxStatus** (závisející na typu poskytovatele služeb), která se rovněž nachází v datovém bloku **ProgramData**.[2]

5.4.3 Pasivní data

Kvůli mnoha limitacím ze strany PLC bylo nutné pasivní část AAS výrazně omezit. V PLC nejsou pasivní data rozděleny do hlavičky a těla. Taktéž není rozlišováno mezi identifikací samotného assetu a AAS. Pasivní data jsou uložena v datovém bloku **Passive AAS**. Jedná se o proměnné datového typu string **ID**, **Name** a **Version**, které jsou napojeny na OPC UA prostředí. Rovněž se v tomto datovém bloku nachází proměnná **Type**, která sice není viditelná pro klienta s přístupem k danému AAS,

[2]Je nezbytné, aby tuto proměnou měnil přímo v místě jejího uložení bez jejího přenosu do parametrů InOut Zařízení. V případě jejího přenosu přes vstupní/výstupní parametry jednotlivých bloků vzniká její nežádoucí přepisování z důvodu implementace aktivního AAS.

Tab. 5.8: Povinné proměnné pro funkčnost OPC UA metod.

Typ parametru	Název	Datový typ	Poznámka
Static	OPC_UA_ServerMethod-Pre_Instance	OPC_UA_ServerMethodPre	Její hodnoty jsou uvedeny v tabulce 5.6.
Static	OPC_UA_ServerMethod-Post_Instance	OPC_UA_ServerMethod-Post	Její hodnoty jsou uvedeny v tabulce 5.7.
Static	statUAM_Status_Pre	"typeOPCUA-Status"	Její hodnoty jsou uvedeny v tabulce 5.9.
Static	statUAM_Status_Post	"typeOPCUA-Status"	Její hodnoty jsou uvedeny v tabulce 5.9.
Static	UAMethod_In-Parameters	Struct	V našem případě se jedná o WString od klienta, neboť TIA portal nepodporuje pouhé stringy pro tyto účely. Jestliže metoda nemá vstupní parametry (případ metody **GetSubmodel**, tak se tato proměnná nedefinuje.
Static	UAMethod_Out-Parameters	Struct	V našem případě se jedná o WString, který vracíme klientovi (TIA portal nepodporuje pouhé stringy pro tyto účely.). Jestliže by metoda neměla výstupní parametry, což se v mé implementaci nestalo, tak by se tato proměnná nedefinovala.
Static	InputArray	"typeArrayMessage"	Statická datová struktura, do které se ukládá zpráva z klienta, rozdělená až do pěti slov.
Temp	tempUAM_MethodHandling	"typeOPCUA-Methodhandling"	V případě potřeby při budoucích úpravách kódu je možné tuto proměnnou přesunout do typu Static.

avšak je využívána pro metodu GetSubmodel. Tato proměnná se rovněž může přidat do OPC UA prostředí a sloužit jako redundance metody GetSubmodel.[3]

5.4.4 PLC datové typy

V rámci programu v PLC byly implementovány vlastní uživatelské datové typy. Tyto speciální datové typy jsou vitální pro správnou funkčnost AAS v PLC. Jsou jimy:

- **typeArrayMessage**: Jedná se o strukturu typu pole o velikosti 5 stringů, do nichž jsou vkládána jednotlivá slova v rámci volání metod pomocí mnou vytvořené funkce **GetMessage**.
- **typeOPCUAStatus**: Jedná se o datový typ obsahující 4 proměnné (viz tabulka 5.9). Je to nezbytný datový typ pro funkčnost knihovních funkcí **OPC_UA_ServerMethodPre** a **OPC_UA_ServerMethodPost**, které jsou v volány v rámci OPC UA metod.

Tab. 5.9: Výčet uživatelského datového typu **typeOPCUAStatus**.

Název	Datový typ
done	Bool
busy	Bool
error	Bool
status	DWord

- **typeOPCUAMethodhandling**: Jedná se o datový typ složený ze tří proměnných (viz tabulka 5.10). Slouží pro indikaci událostí metod včetně jejich výsledků.

Tab. 5.10: Výčet uživatelského datového typu **typeOPCUAMethodhandling**.

Název	Datový typ
UAMethod_Called	Bool
UAMethod_Result	DWord
UAMethod_Finished	Bool

[3]Pro přídání do OPC UA prostředí a umožnění jejího čtení pro klienta je nutné přidat tuto proměnnou jako nový uzel v SiOME nebo jinému programu, která dovoluje vhodně upravovat jmenný prostor OPC UA a následně upravený .xml soubor opět importovat do TIA portalu v sekci **OPC UA communication/Server interfaces**, viz návod v podkapitole 5.5.2 TIA Portal.

5.4.5 PLC tagy

Tabulka **OPC UA Tag table** byla importována ze standardizovaného .xml souboru, jež je dostupný na [2]. Tato tabulka, nacházející se v záložce **User constants**, obsahuje veškeré statusy (chyby), které mohou při provozu AAS nastat. Jsou datového typu DWord a každá proměná má mimo jiné komentář obsahující její základní popis.

Je zde možnost tyto standardizované statusy nevyužít a vytvořit si vlastní, avšak to by mohlo vyvolat negativní účinky v případných nástavbách kódu, aktualizacích firmwaru či komunikacemi se zařízeními, které by byly vytvořeny s ohledem na tuto standardizaci. Z těchto důvodů se doporučuje zmíněnou možnost využít a standardizovanou tabulku do prostředí TIA portal naimportovat.

5.5 Mapování na OPC UA

V následujících podkapitolách je popsáno mapování OPC UA prostředí do TIA portalu včetně jeho tvorby v programu SiOME.

5.5.1 SiOME

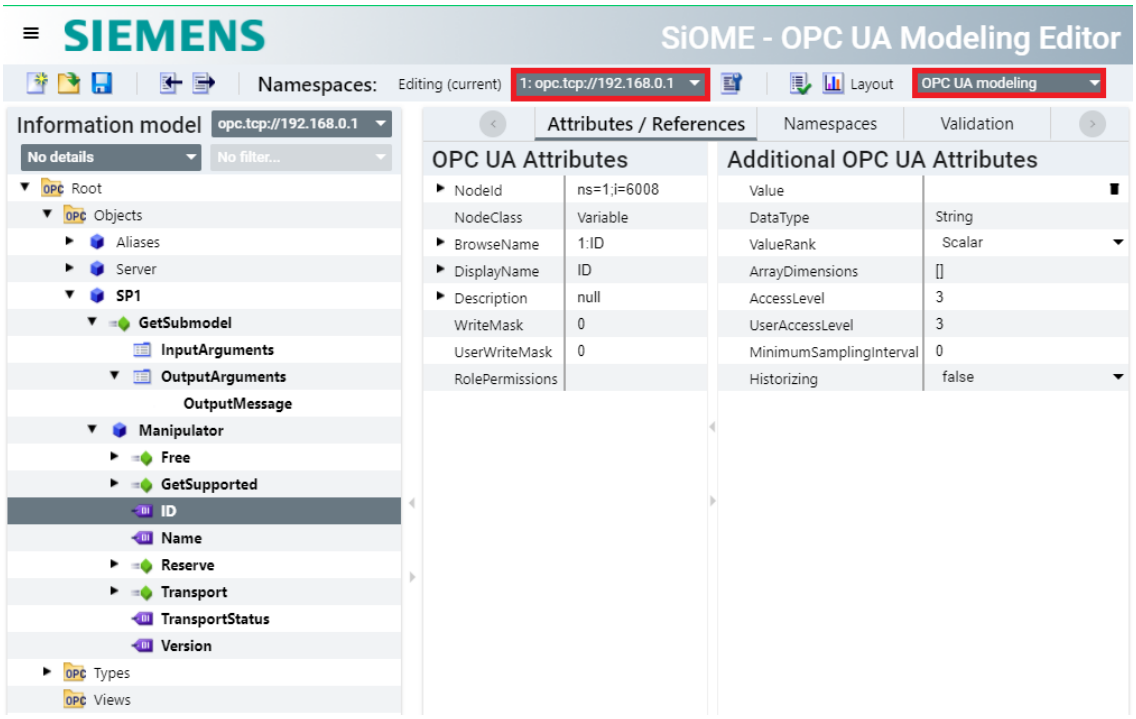
Na propojení bylo vytvořeno vlastní OPC UA prostředí formou .xml souboru pomocí programu SiOME (Siemens OPC UA Modelling Editor). Tento program umožňuje měnit OPC UA uzly a taktéž jeho přímé propojení na TIA portal[4]. Pro úpravu .xml souboru je nutné zvolit v pravém horním rohu **OPC UA modelling** a uprostřed horní sekce zvolit vhodný **Namespace**. Poté lze přidávat OPC UA metody, nastavovat jejich vstupní a výstupní parametry a přidávat pasivní data. Toto se provádí pravým kliknutím na požadovaném místě uzlu (levá část obrazovky) a zvolení **Add Instance**. Následně lze vybrat, zdali se bude jednat o objekt, metodu, či pasivní data. Kromě uživatelem vytvořené OPC UA struktury AAS jsou zde i objekty jmenného prostoru, které jsou nezbytné k správné funkčnosti základních funkcí OPC UA. Tyto objekty se vytváří automaticky prostředím SiOME.

Na obrázku 5.2 vidíme strukturu programu včetně nastavení vlastností proměnné ID v pravé části okna.

5.5.2 TIA Portal

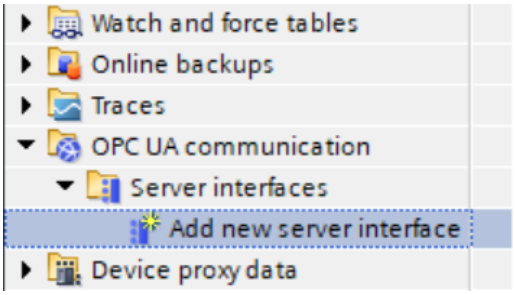
Propojení metod a pasivních dat se provádí v sekci **OPC UA communication/Server interfaces**. Zde se přidalo nové rozhraní, zvolilo **Companion specifi-**

[4]Přímé propojení s TIA portalem není podporováno pro operační systém Windows 10 Home. Je nutná verze Professional.



Obr. 5.2: Tvorba OPC UA prostředí v programu SiOME.

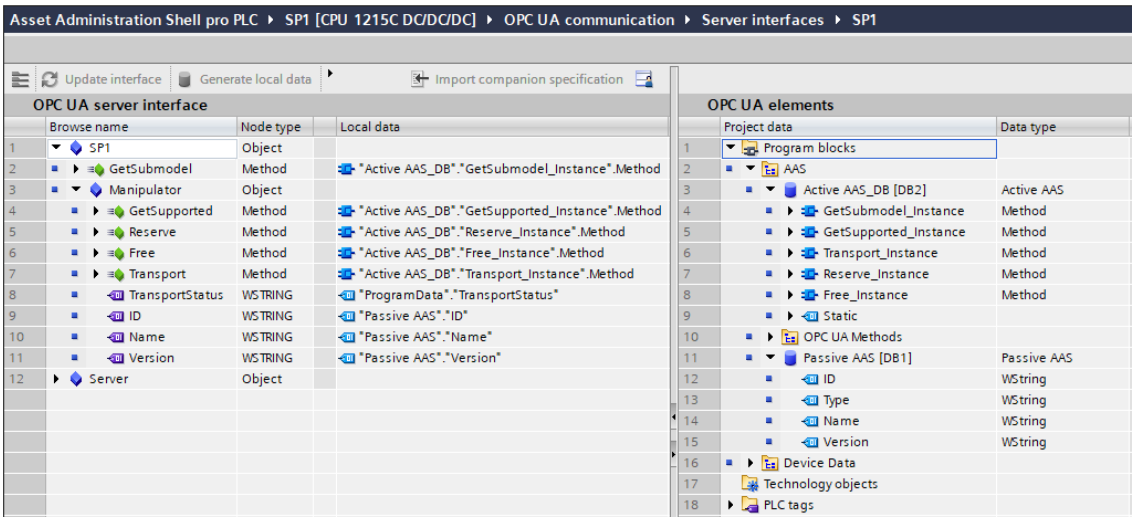
cation a byl vybrán .xml soubor ze SiOME, zobrazeno na obrázku 5.3.



Obr. 5.3: Přidání definovaného OPC UA prostředí do TIA portalu.

Po otevření rozhraní OPC UA serveru jsou k dispozici 2 objekty: Samotné prostředí pro mapování metod (v mé implementaci nazváno SP1-3) a následně objekt Server, který obsahuje popis jmenného prostoru, zobrazeno na obrázku 5.4.

V průběhu mapování je okno rozděleno na dvě části: **OPC UA server interface** a **OPC UA elements**. Mapování se provádí přetáhnutím proměnných a metod z části **Project data** nacházejících se v oblasti **OPC UA elements** do sloupce **Local data**, který je na levici. Vytvořené Metody a pasivní data z části **Project data** jsou zobrazeny pod příslušnými datovými bloky. Jestliže tam daná metoda



Obr. 5.4: Propojení dat AAS z TIA portalu do OPC UA prostředí.

není zobrazena, svědčí to o špatném použití knihovních funkcí **OPC_UA_ServerMethodPre** a nebo **OPC_UA_ServerMethodPost**. V mé implementaci jsou zobrazeny pod datovým blokem **Active AAS**, neboť jsou v něm uloženy jednotlivé funkční bloky ve formě multi-instance.

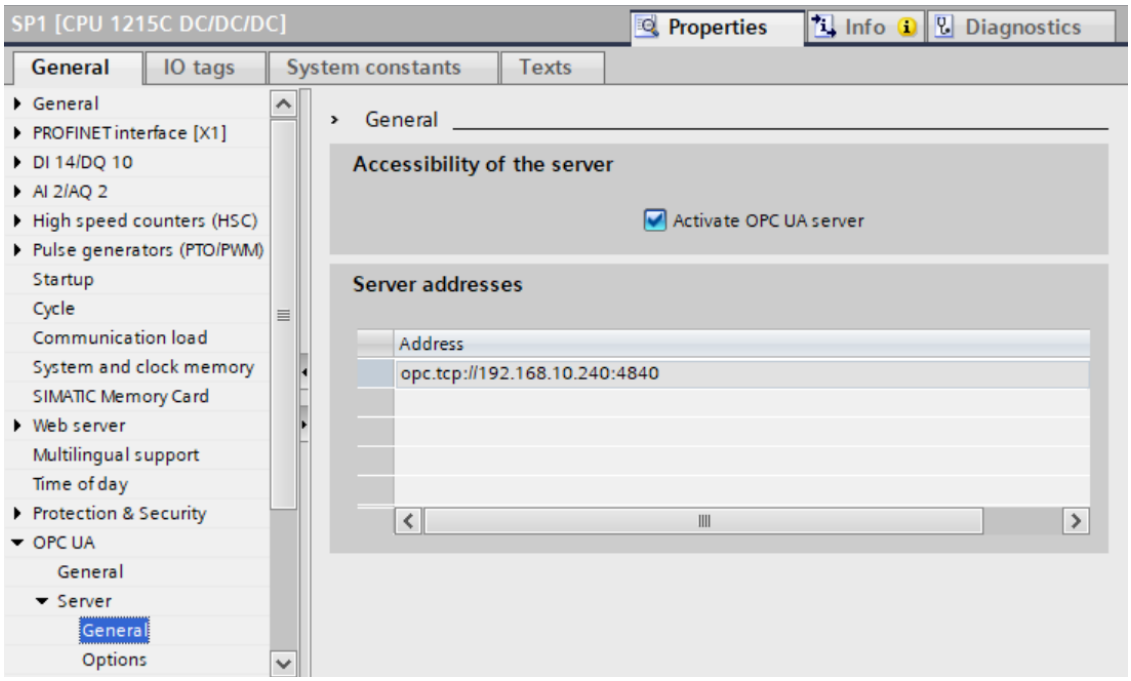
5.6 Volání metod

Pro sprovoznění serveru je nutné aktivovat OPC UA server v záložce **Device configuration/OPC UA/General**, kde se rovněž nachází jeho adresa pro připojení klientů, zobrazeno na obrázku 5.5.

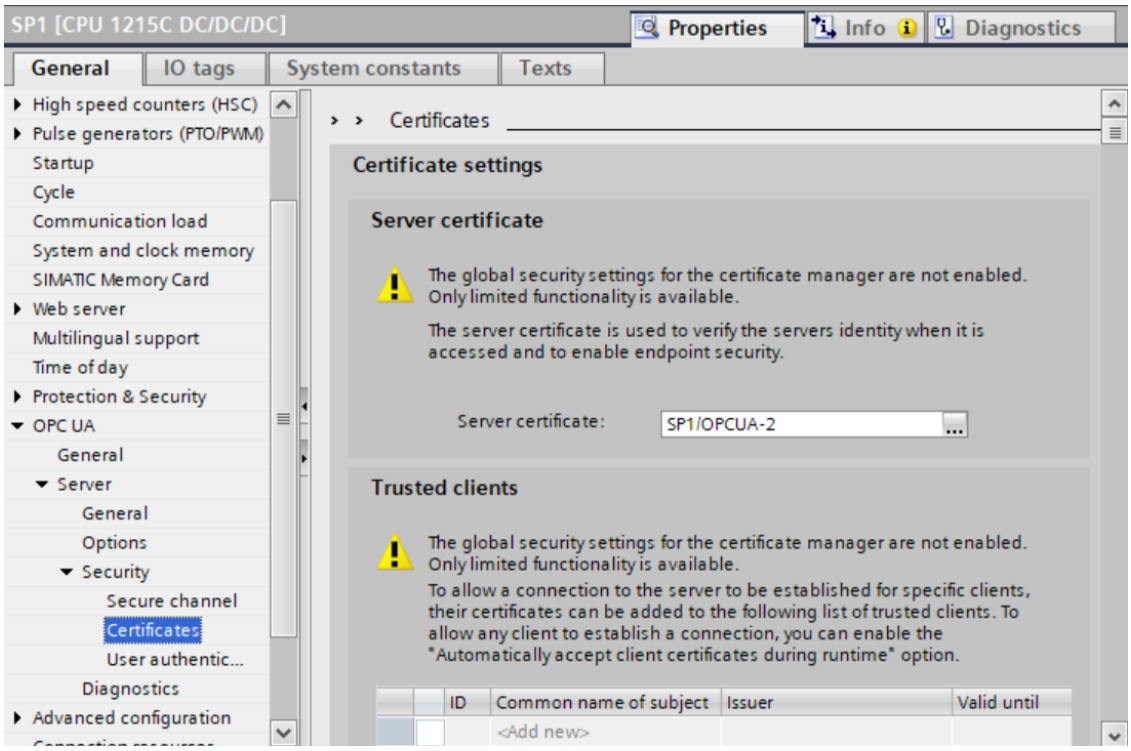
Následně je nutné zvolit jeho certifikát v záložce **Device configuration/OPC UA/Security/Certificates**, zobrazeno na obrázku 5.6.

Z důvodu bezpečného chodu PLC a znemožnění přístupu na server neoprávněným entitám se doporučuje nastavit **Trusted clients** ve stejné záložce. V mém kódu je toto zabezpečení vypnuto z důvodu lepší dostupnosti pro široké použití, a tedy neprobíhá autentizace klientů, kteří se připojují na server. Komunikace se zabezpečením byla v rámci mé práce taktéž testována a fungovala bez problémů.

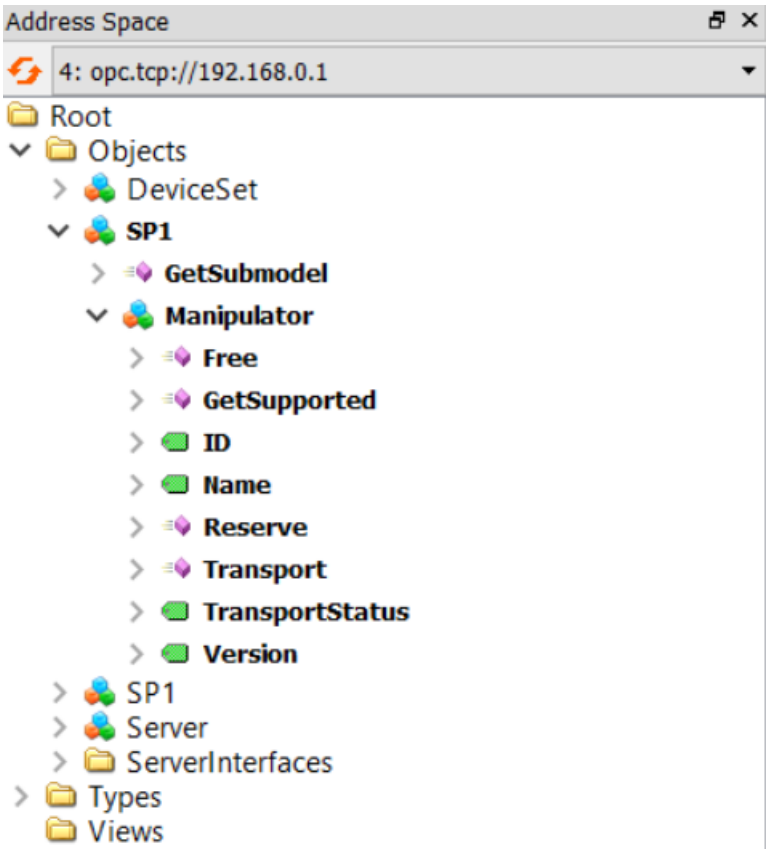
Pro připojení k serveru je nutný OPC UA klient. V mé práci byl využit **Ua Expert V1.6.1** kvůli jeho široké podpoře a dostupnosti. V tomto prostředí byl přidán nový server v sekci **Project**, zadána jeho adresa z PLC ve formátu **opc.tcp://IP ADRESS:PORT** a v závislosti na zvoleném zabezpečení proběhlo připojení. V levé části se po úspěšném připojení objeví OPC UA prostředí serveru. Pro lepší orientaci je možno zvýraznit jmenný prostor (namespace) vytvořený v SiOME.



Obr. 5.5: Aktivace OPC UA serveru na PLC.



Obr. 5.6: Aktivace OPC UA licence na PLC.



Obr. 5.7: Zobrazení OPC UA prostředí v programu UaExpert.

Na obrázku 5.7 můžeme pozorovat strukturu OPC UA prostředí. Zařízení SP1 má metodu **GetSubmodel** a objekt **Manipulator**. Tento objekt má dále 4 metody, 3 pasivní data reprezentující pasivní část AAS a proměnnou **TransportStatus**, která je namapována totožným způsobem jako pasivní data. Ta se mění v důsledku volání metody **Transport** a samotného zařízení, které přes ni informuje jeho aktuální stav. Metody se v prostředí UaExpert volají pravým kliknutím na danou metodu a následně na **Call...**. Proměnné lze zobrazit jejich přetažením do okna **Data Access View**, kde je možnost mimo jiné vidět časové známky jejich zdrojů. Taktéž tam lze upravit širokou škálu nastavení aktualizace jednotlivých proměnných.[5]

Na obrázku 5.8 je názorný příklad komunikace mezi AAS za pomoci mé implementace. Skládá se ze 3 poskytovatelů služeb a jednoho žadatele služeb. Pro tento zjednodušený příklad je předpoklad již existujícího výrobku (skleničky) v testbedovi Barman, který se aktuálně snaží dostat do sféry vlivu dávkovače, který má za cíl přidat do skleničky materiál 1 (voda) v množství 3 (ml). Nejdříve si z dostupných

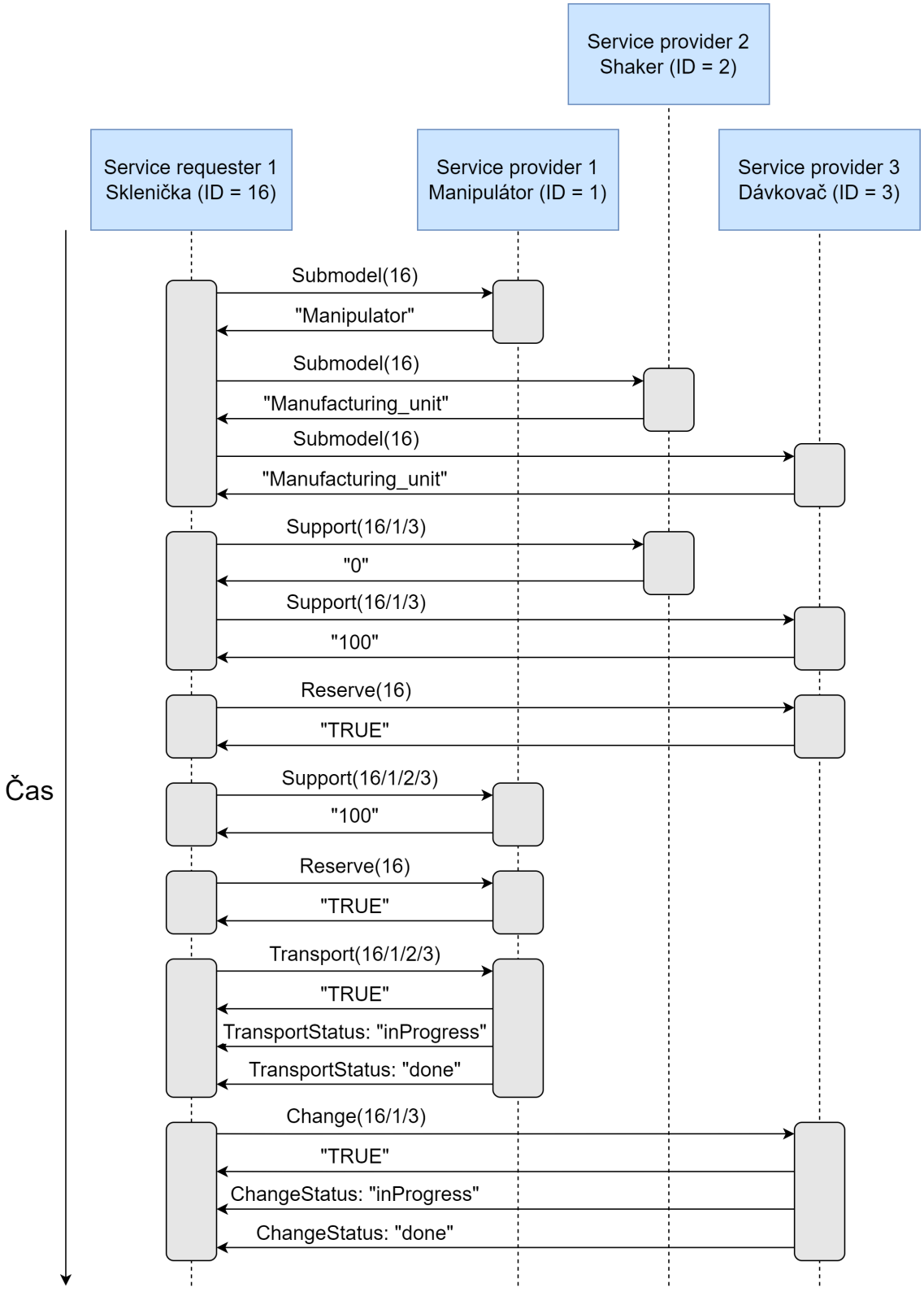
[5]Jestliže se u pasivního AAS nezobrazují hodnoty a **Datatype** je NULL, pak je s největší pravděpodobnostní špatně nastavena aktualizace jejich hodnot na straně klienta. Tato situace se typicky projevuje při restartu PLC. Další možností je pak špatně nastavený datový typ z SiOME.

zařízení zjistí, kterého typu jsou, následně se shakeru a dávkovače zeptá, zdali poskytují danou službu. Zvolí si tu nejvhodnější (v tomto případě i jedinou) nabídku od dávkovače, zarezervuje se u něj, takže dávkovač nadále nemůže obsluhovat jiné žadatele služeb, dohodne si transport s manipulátorem do sféry vlivu dávkovače a podle proměnných **TransportStatus** a **ChangeStatus** ví, v jakém stavu se aktuálně nachází. Příklad volání metody je ukázán na obrázku 5.9.

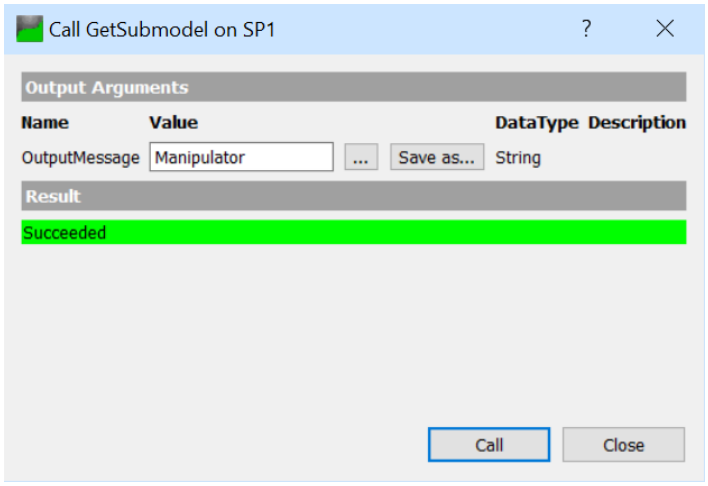
5.7 Testování

Vzhledem k chybějící implementaci SR byla funkcionalita AAS ověřena manuálně. Za pomoci přiloženého programu byla vytvořena síť pěti virtuálních PLC řady S7 - 1500 ve složení 1 Storage, 2 Manipilator a 2 Manufacturing_unit, které simulovaly malou továrnu. V PLC programech jim taktéž byly přiřazeny odlišné parametry v pasivním AAS a rozdílnou nabídku jejich služeb v rámci aktivního AAS. S ohledem na prověření bezpečnosti byla taktéž využita možnost zabezpečení PLC, kdy byla zakázána **Guest authentication** a přihlášovací údaje byly pořadovány od klientů, což lze v přiloženém programu najít v sekci **Device configuration/OPC UA/Server/Security/User authentication**, viz obrázek 5.10.

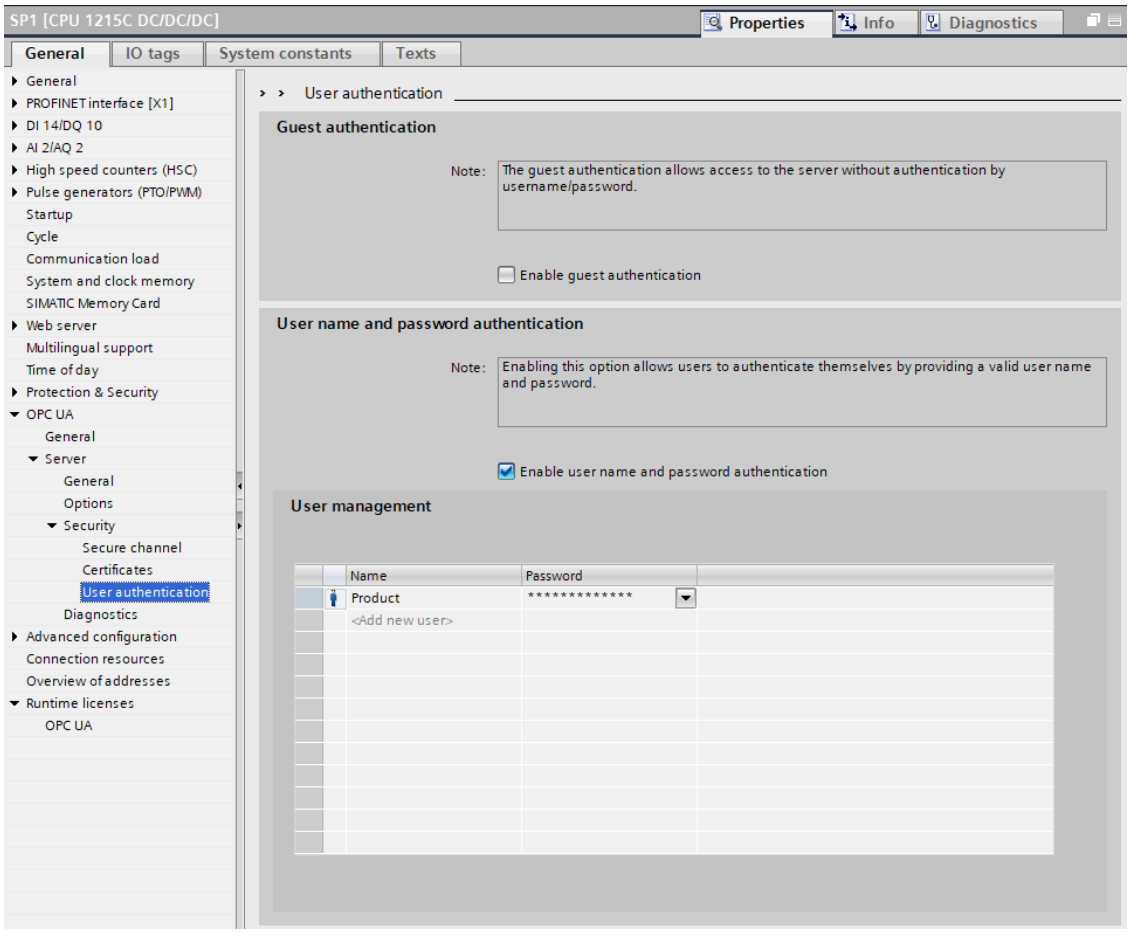
Následně byla inicializována spojení za pomoci klienta UaExpert, kde bylo otestováno, že bez znalosti přihlašovacích údajů se není možné na server připojit. Klienti v tomto případě reprezentovali produkty ve výrobě, které volaly metody žadatelů služeb s cílem splnit jejich individuální výrobní postupy. Reagovali na odpovědi volaných metod a vyhodnocovali další postupy. Jednalo se tedy o daleko komplikovanější diagram nežli ten na obrázku 5.8. Při testování se potvrdilo, že se PLC chovají dle očekávání. Taktéž bylo zjištěno, že z důvodu chybějících rezervačních front nastává při větším množství produktů k méně efektivní výrobě, což znemožňuje případnou priorizaci výrobků.



Obr. 5.8: Příklad komunikace AAS za použití metod.



Obr. 5.9: Volání AAS metody v prostředí UaExpert.



Obr. 5.10: Nastavení zabezpečeného připojení PLC.

6 Budoucí práce

V rámci mé dosavadní implementace nebyly některé aspekty AAS v PLC plně dořešeny.

6.1 Service Requester

Prvním nedostatkem je již zmíněný chybějící Service Requester, který znemožňuje mé řešení aktuálně implementovat do praxe. Musí mít schopnost komunikace jako klient v rámci OPC UA. Dále musí ctít již vytvořené standardy ohledně aktivního a pasivního AAS a musí být zajištěna jeho bezpečnost, aby bylo sníženo riziko sabotáže přes falešného či upraveného žadatele služeb. Pro žadatele služeb je implementace do PLC naprosto nevhodná a pro její budoucí implementaci se doporučuje vydat cestou například programu v C++ či Pythonu, aby se vyhnulo již zmiňovaným limitacím PLC.

6.2 Rezervační fronty a priorizace výrobků

Druhým nedostatkem je taktéž již zmíněná chybějící rezervační fronta pro žadatele služeb v PLC. Její aplikace by kromě plynulejší výroby umožnila taktéž priorizaci služeb pro výrobky s vyšší prioritou typicky kvůli většímu finančnímu ohodnocení za produkt či nutnosti jeho rychlé výroby. Toto by šlo provést ze strany serveru, který by po dohodě s žadateli služeb upravoval svou rezervační frontu s cílem preference výrobků s vyšší prioritou, která by se mohla přidat do dalšího slova vstupního parametru metody Reserve tak, aby obsahoval **SR ID/priorita**.

Při implementaci rezervačních front a s tím související priorizaci výrobků je nutné dodržení Real-Time komunikace, což zajišťuje OPC UA protokol. Taktéž je nutné zvýšit nároky na zařízení, které by nyní muselo vědět čas nutný pro splnění daných služeb. Tento čas, který by měl být udáván převážně v ms, by bylo nutné buďto zakomponovat do metody GetSupported, nebo ho využít při tvorbě nové metody, která by sloužila k vyhledávání nejbližšího časového okna s parametry jeho začátku a délky či konce. Metoda Reserve by se rovněž musela obohatit o zmíněné parametry. S ohledem na již zaběhnutý standard ve formě stringů reprezentující slova oddělená znaky / se doporučuje pokračovat v tomto standardu a pouze přidat další slova do již vytvořeného vstupního parametru.[1]

[1]Pro PLC řady S7-1200 není podporován datový formát času LDT, který je nezbytný pro přenos časových formátů v rámci OPC UA metod. S ohledem na tuto skutečnost se opět doporučuje využít parametr typu string (v PLC WString).

Výrobek by si tedy metodou GetSupported zjistil potřebný čas na jeho službu u SP, následně by si nově vytvořenou metodou zjistil nejbližší časový úsek o daném trvání a rezervoval by se na tuto dobu do fronty poskytovatele služeb. V případě, že by následně výrobek s vyšší prioritou požadoval služby stroje, by server výrobek s nižší prioritou odregistroval a obeznámil mu tuto skutečnost přes příslušnou proměnnou v OPC UA prostředí. Pouhé posunutí vykonání služby v čase se nedoporučuje, neboť by na tuto dobu výrobek danou službu nemusel potřebovat. Jednalo by se tedy o takové výrobní procesy, kde jsou služby na sobě navzájem závislé (typicky Manipulator - Manufacturing_unit - Manipulator).

Kromě priorizace výrobků by implementace rezervačních front taktéž umožnila prediktivní údržbu, optimalizaci energetických spotřeb a lepší možnosti uplatnění AAS pro továrny s větším množstvím provázaných technologických procesů.

7 Závěr

V rámci bakalářské práce byly vysvětleny důležité pojmy v dané problematice jako jsou I4.0, RAMI 4.0, AAS a podobně. Dále bylo sepsáno desatero požadavků pasivního AAS a vývojové diagramy aktivní části AAS pro následné implementace datových modelů a základní funkcionality AAS do PLC. Z desatera požadavků pasivního AAS je patrné množství překážek pro implementaci AAS do PLC. Byly zmíněny požadavky na příslušné normy a zdůvodněno jejich případné nedodržení.

Byly popsány dvě nejvýraznější komunikační platformy pro AAS v PLC. Jednalo se o OPC UA a MQTT. Tyto komunikační protokoly byly popsány a jejich přednosti byly vyzdvihnuty s ohledem na AAS v PLC. Byl vybrán OPC UA pro následnou implementaci z důvodu jeho výrazných výhod v oblastech Real-Time komunikace a umožnění volání metod.

V rámci praktické části byla AAS síť testována ve virtuálním prostředí na PLC řady S7-1500 a následně přenesena na reálné PLC řady S7-1200. Pro implementaci byly využity OPC UA metody, které byly voleny s ohledem na jejich univerzálnost a praktičnost v reálné továrně. Tyto metody jsou v rámci mé bakalářské práce detailně popsány s cílem jejich replikace do praxe ze třetích stran. OPC UA prostředí bylo modelováno v programu SiOME, které bylo následně napojeno na program v TIA portalu. Volání metod a celkové testování funkčnosti bylo kvůli chybějící implementaci SR prováděno manuálně přes UaExpert. Nad rámec zadání bakalářské práce byla zpracována kapitola 6 Budoucí práce, kde bylo představeno řešení částí AAS, které nebyly cílem této bakalářské práce, avšak jsou vhodné pro implementaci AAS do PLC v praxi.

Má implementace představuje nový přístup k řízení v průmyslu a přesto nevyvíjí nároky na nový hardware kromě již standardních PLC. Představuje možnost aplikovat toto decentralizované řízení na již stávající výrobní jednotky v průmyslu a vyzdvihuje jeho možné uplatnění v praxi.

Literatura

[1] Functional View of the Asset Administration Shell in an Industrie 4.0 System Environment [online]. Plattform Industrie 4.0, Duben 2021, [cit. 2021-09-27].
Dostupné z: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Functional-View.pdf?__blob=publicationFile&v=5

[2] *OPC UA methods for the SIMATIC S7-1500 OPC UA server [online]*. Leden 2022, [cit. 2022-02-14].
Dostupné z: https://support.industry.siemens.com/cs/document/109756885/opc-ua-methods-for-the-simatic-s7-1500-opc-ua-server-?dti=0&lc=en-WW

[3] Adolphs, P.; Auer, S.; Bedenbender, H.; aj.: Structure of the Administration Shell: Continuation of the Development of the Reference Model for the Industrie 4.0 Component [online]. *ZVEI and VDI, status report*, Duben 2016, [cit. 2021-10-17].
Dostupné z: https://www.academia.edu/35536663/Structure_of_the_Administration_Shell_Continuation_of_the_Development_of_the_Reference_Model_for_the_Industrie_4.0_Component.pdf

[4] Adolphs, P.; Auer, S.; Bedenbender, H.; aj.: The Structure of the Administration Shell: Trilateral perspectives from France, Italy and Germany [online]. Technická zpráva, Duben 2018, [cit. 2021-09-29].
Dostupné z: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.pdf?__blob=publicationFile&v=4

[5] Arm, J.; Benesl, T.; Marcon, P.; aj.: Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0 [online]. *Sensors*, ročník 21, Březen 2021: str. 2004, doi:10.3390/s21062004, [cit. 2021-06-01].
Dostupné z: https://www.researchgate.net/publication/350068136_Automated_Design_and_Integration_of_Asset_Administration_Shells_in_Components_of_Industry_40

[6] Bader, S.; Barnstedt, E.; Bedenbender, H.; aj.: Details of the Asset Administration Shell. Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01) [online]. Technická zpráva, Listopad 2020, [cit. 2021-06-01].
Dostupné z: https://www.plattform-i40.de/IP/Redaktion/EN/

Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html

[7] Bader, S.; Berres, B.; Boss, B.; aj.: Details of the Asset Administration Shell. Part 2 -Interoperability at Runtime - Exchanging Information via Application Programming Interfaces (Version 1.0RC01) [online]. Technická zpráva, Listopad 2020, [cit. 2021-09-27].
Dostupné z: https://www.researchgate.net/publication/346606788_Details_of_the_Asset_Administration_Shell_Part_2_-Interoperability_at_Runtime_-_Exchanging_Information_via_Application_Programming_Interfaces_Version_10RC01

[8] Bader, S. R.; Maleshkova, M.: The Semantic Asset Administration Shell [online]. In *Semantic Systems. The Power of AI and Knowledge Graphs*, editace M. Acosta; P. Cudré-Mauroux; M. Maleshkova; T. Pellegrini; H. Sack; Y. Sure-Vetter, Cham: Springer International Publishing, Listopad 2019, ISBN 978-3-030-33220-4, s. 159–174, [cit. 2021-12-01].
Dostupné z: https://link.springer.com/chapter/10.1007/978-3-030-33220-4_12

[9] Bedenbender, H.; Bentkus, A.; Epple, U.; aj.: Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation [online]. Červen 2017, [cit. 2021-12-01].
Dostupné z: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2017/Juni/Industrie_4.0_Plug_and_produce/Industrie-4.0-_Plug-and-Produce-zvei.pdf

[10] Bedenbender, H.; Bentkus, A.; Epple, U.; aj.: Relationships between I4. 0 Components-Composite components and smart production [online]. Červen 2017, [cit. 2021-10-23].
Dostupné z: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/relationships-i40-components.pdf?__blob=publicationFile&v=5

[11] Belyaev, A.; Diedrich, C.: Specification "Demonstrator I4.0-Language"v3.0 [online]. Technická zpráva, Červenec 2019, [cit. 2021-10-14].
Dostupné z: https://www.researchgate.net/publication/334429449_Specification_Demonstrator_I40-Language_v30

[12] Bradac, Z.; Marcon, P.; Zezulka, F.; aj.: Digital Twin and AAS in the Industry 4.0 Framework [online]. *IOP Conference Series: Materials Science and Engineering*, ročník 618, Říjen 2019, doi:10.1088/1757-899X/618/1/012001,

[cit. 2021-10-6].
Dostupné z: https://www.researchgate.net/publication/336880479_Digital_Twin_and_AAS_in_the_Industry_40_Framework

[13] Diesner, M.; Pfrommer, J.; Rauschecker, U.; aj.: Industrie 4.0 Begriffe/Terms [online]. *VDI status report Industrie 4.0*, Duben 2017, [cit. 2021-10-13].
Dostupné z: https://www.researchgate.net/publication/317840505_Industrie_40_BegriffeTerms

[14] Dorst, W.; Glohr, C.; Diegner, B.; aj.: Implementation Strategy Industrie 4.0: Report on the Results of the Industrie 4.0 Platform [online]. Leden 2016, [cit. 2021-10-24].
Dostupné z: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/Implementation_Strategy_Industrie_4.0_-_Report_on_the_results_of_Industrie_4.0_Platform/Implementation-Strategy-Industrie-40-ENG.pdf

[15] Dönicke, N.; Gamer, T.; Heer, T.; aj.: Security der Verwaltungsschale [online]. Duben 2017, [cit. 2021-10-24].
Dostupné z: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/security-der-verwaltungsschale.pdf?__blob=publicationFile&v=6

[16] Dürst, M.; Suignard, M.: Internationalized Resource Identifiers (IRIs) [online]. Leden 2005, [cit. 2021-10-18].
Dostupné z: https://www.researchgate.net/publication/200034559_Internationalized_Resource_Identifiers_IRIs

[17] Haag, S.; Anderl, R.: Digital twin – Proof of concept [online]. *Manufacturing Letters*, ročník 15, Leden 2018: s. 64–66, ISSN 2213-8463, doi: https://doi.org/10.1016/j.mfglet.2018.02.006, [cit. 2021-10-06].
Dostupné z: https://www.sciencedirect.com/science/article/pii/S2213846318300208

[18] Madiwalar, B.; Schneider, B.; Profanter, S.: Plug and Produce for Industry 4.0 using Software-defined Networking and OPC UA [online]. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Září 2019, s. 126–133, [cit. 2021-12-22].
Dostupné z: https://www.researchgate.net/publication/335078956_Plug_and_Produce_for_Industry_40_using_Software-defined_Networking_and_OPC_UA

[19] Marcon, P.; Diedrich, C.; Zezulka, F.; aj.: The Asset Administration Shell of Operator in the Platform of Industry 4.0 [online]. In *2018 18th International Conference on Mechatronics - Mechatronika (ME)*, Prosinec 2018, s. 1–5, [cit. 2021-06-01].
Dostupné z: https://ieeexplore.ieee.org/abstract/document/8624699

[20] Profanter, S.; Dorofeev, K.; Zoitl, A.; aj.: OPC UA for plug & produce: Automatic device discovery using LDS-ME [online]. In *2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA)*, IEEE, Září 2017, s. 1–8, [cit. 2021-12-22].
Dostupné z: https://www.researchgate.net/publication/317358980_OPC_UA_for_plug_produce_Automatic_device_discovery_using_LDS-ME

[21] Wagner, C.; Grothoff, J.; Epple, U.; aj.: The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant [online]. Září 2017, s. 1–8, doi:10.1109/ETFA.2017.8247583, [cit. 2021-06-01].
Dostupné z: https://www.researchgate.net/publication/322324072_The_role_of_the_Industry_40_asset_administration_shell_and_the_digital_twin_during_the_life_cycle_of_a_plant

[22] Wenger, M.; Zoitl, A.; Müller, T.: Connecting PLCs with their asset administration shell for automatic device configuration [online]. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, IEEE, Červenec 2018, s. 74–79, [cit. 2021-10-01].
Dostupné z: https://ieeexplore.ieee.org/document/8472022

[23] Ye, X.; Hong, S.: Toward the Plug-and-Produce Capability for Industry 4.0: An Asset Administration Shell Approach [online]. *IEEE Industrial Electronics Magazine*, ročník 14, Prosinec 2020: s. 146–157, [cit. 2021-12-01].
Dostupné z: https://www.researchgate.net/publication/347485386_Toward_the_Plug-and-Produce_Capability_for_Industry_40_An_Asset_Administration_Shell_Approach

Seznam symbolů a zkratek

AAS	Asset Administraton Shell
BITKOM	Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.
ČR	Česká republika
DT	Digital Twin - Digitální dvojče
I4.0	Industry 4.0 - Průmysl 4.0
IRDI	International Registration Data Identifier
LDS ME	Local Discovery Server with Multicast Extension
MQTT	Message Quering Telemetry Transport
OPC UA	Open Platform Communications United Architecture
PLC	Programmable Logic Controller - Programovatelný Logický Automat
RAMI	Reference Architecture Model for Industre 4.0 - Referenční architektonický model pro průmysl 4.0
SGAM	Smart Grid Architecture Model (128 bit)
SiOME	Siemens OPC UA Modeling Editor
UaExpert	Unified Automation Expert
URI	Uniform Resource Identifier - jednotný identifikátor zdroje
UUID	University Unique IDentifier
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e.V.
VDI	Verein Deutscher Ingenieure e.V
VDMA	Verband Deutscher Maschinen und Anlagenbau e.V.
ZVEI	Zentralverband Elektrotechnik und Elektronikindustrie e.V.