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 OPERÁTORA

ASSET ADMINISTRATION SHELL FOR THE OPERATOR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Vojtěch Houdek

AUTHOR

VEDOUCÍ PRÁCE — Ing. Ondřej Baštán

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: Vojtěch Houdek ***ID:*** 221332

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

NÁZEV TÉMATU:

Asset Administration Shell pro operátora

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je seznámit se s funkcionalitou a standardy pro AAS a následně vytvořit operátorské rozhraní pro začlenění lidské obsluhy do řetězce výroby.

1. Seznamte se s pojmem AAS.
2. Definujte a popište funkce pasivní části AAS.
3. Definujte a popište základní funkce aktivní části (včetně vývojových/flow diagramů)
4. Vyberte vhodný komunikační protokol a platformu pro operátorské AAS.
5. Implementujte datový model AAS pro zvolenou platformu.
6. Implementujte základní funkcionalitu aktivní části AAS na zvolené platformě.
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. Ondřej Baštán

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

ABSTRAKT

Tato bakalářská práce se zabývá tématem AAS (asset Administration shell), tedy prvkem průmyslu 4.0. První polovina práce je tvořena teoretickým rozborem pojmu AAS a jeho aktivní a pasivní částí. Druhá polovina je věnována praktickému navrhu a implementace AAS pro operátora. Je zde řešena implementace komunikační architektury a volba vhodného Hardwaru pro AAS. Výsledná realizace se zabývá programováním aktivní a pasivní částí AAS v C/C++ a mobilní aplikací v Android Studiu za pomocí programovacího jazyka Kotlin.

KLÍČOVÁ SLOVA

AAS, Průmysl 4.0, pasivní část AAS, aktivní část AAS, OPC UA, C++, Kotlin, Android.

ABSTRACT

This bachelor thesis deals with the topic of AAS (asset administration shell), an element of industry 4.0. The first half of the work consists of a theoretical analysis of the concept of AAS and its active and passive parts. The second half is devoted to the practical design and implementation of AAS for the operator. The implementation of the communication architecture and the selection of suitable Hardware for AAS are addressed here. The resulting implementation deals with programming the active and passive part of AAS in C/C ++ and mobile applications in Android Studio using the programming language Kotlin.

KEYWORDS

AAS, Industry 4.0, passive part of AAS, active part of AAS, OPC UA, C++, Kotlin, Android

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

HOUDEK, Vojtěch. *ASSET ADMINISTRATION SHELL PRO OPERÁTORA*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 72 s. Bakalářská práce. Vedoucí práce: Ing. Ondřej Baštán

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Vojtěch Houdek
VUT ID autora:	221332
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	ASSET ADMINISTRATION SHELL PRO OPERÁTORA

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. Ondřeji Baštánovi, za odborné vedení, konzultace, velkou míru trpělivosti a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Tato bakalářská práce se věnuje problematice komplexního pojmu asset administation shell (AAS) pocházející z prostředí průmyslu 4.0 (I4.0). Protože se jedná o nový a doposud nestandardizovaný prvek v I4.0, je zapotřebí se seznámit s dosavadními postupy a pokusy o definování AAS, a jeho částí. Před praktickou realizací AAS, je proto nutné nejprve navrhnout vlastní přístupy k řešení dané problematiky.

Jeden z cílů této práce je definování pasivní a aktivní části AAS. Tento cíl je vypracován v úvodním teoretickém rozboru, jenž krom definic částí AAS, obsahuje také definice důležitých pojmů z I4.0 pro pochopení kontextu AAS v průmyslovém využití.

Po teoretickém rozboru následuje praktická realizace AAS pro operátora. Při této realizace je zprvu proveden návrh přístupu k důležitým částem AAS, jako jsou datové a komunikační vlastnosti AAS. Tento návrh je proveden ve snaze docílit kompatibility s jinými formami implementací AAS odpovídající vznikajícímu školnímu standardu zabývající se právě AAS.

Implementace navržených přístupů je následně prováděna na vybraných nejvhodnějších platformách softwaru (SW) i hardwaru (HW), tak aby bylo docíleno co možná nejvíce univerzálnosti a praktičnosti řešení.

Výsledná realizace aplikace AAS pro operátora je následně podrobována zkušebnímu testování funkčností, z něhož jsou vyvozeny závěry týkající se úspěšnosti implementace AAS.

Pro usnadnění práce s výslednou realizací AAS byl sepsán pro operátora návod na obsluhu grafického rozhraní odpovídající kompatibilním vlastnostem s ostatními implementacemi AAS.

1 Teoretický rozbor

1.1 Pojem asset administration shell (AAS)

Před řešením problematiky pojmu AAS, je nutné se seznámit s pojmem I4.0, kterého je AAS součástí. Z důvodu komplexnosti tohoto tématu je nutné stručné seznámení s klíčovými pojmy, řadou funkčních principů a příkladů zařízení v rámci I4.0. Také se bude zmíněna práce s daty v I4.0 neboť AAS z velké části řeší právě tento úkon v mnoha podobách.

I4.0 je poměrně rozsáhlá problematika, která se nedá krátce a zároveň podrobně vysvětlit, a proto nadcházející tři stránky budou obsahovat pouze základní představení několika klíčových pojmů v I4.0, pro následné lepší zasazení AAS do jeho kontextu.

1.1.1 Obecný úvod do I4.0

Pojem I4.0 byl poprvé představen na veletrhu v Hannoveru v roce 2013. Označení 4.0 má představovat referenci na čtvrtou průmyslovou revoluci, někteří odborníci ovšem I4.0 považují spíše za přirozenou evoluci třetí průmyslové revoluce. Dalo by se říci, že každou průmyslovou revolucí se z výrobního procesu určitým způsobem vyjímá lidský faktor. V první průmyslové revoluci se nejtěžší lidská práce nahradila parními stroji. Ve druhé průmyslové revoluci se do celého procesu integrovala elektrická energie a tím rozšířila možnosti dalšího pracovního úkonu strojem. Třetí průmyslová revoluce přinesla jednoduché řízení elektrických strojů pomocí elektrického prvku. Ve čtvrté průmyslové revoluci se lze bavit o dalším nahrazením člověka na úrovni řízení výrobních celků a datových zpracování výstupních dat z nich. [20]

I4.0 proto spojuje mnoho nových technologií a pracovních postupů do jednoho celku za účelem vytvoření tzv. chytré továrny. V dnešní době se kompletně chytré továrny v přesném slova smyslu zatím nevyskytují. Představa kompletně chytré továrny je plně autonomní továrna schopna soběstačné realizace výroby námi zadaného úkolu a to od dodavatele přes výrobu až po export. AAS, které je stěžejním tématem této práce, je jedním z důležitých částí možného modelu chytré továrny. [1]

Hlavní páteří I4.0 jsou dozajista kyberfyzické systémy, které realizují jednotlivé části výrobního procesu. Pod tímto pojmem si lze představit technické zařízení, které pracuje na základě pokynů řídícího algoritmu. [2]

Principy I4.0

Jako stěžejní faktor v I4.0 se dá rozhodně považovat digitalizace. Od digitalizace výrobních a obchodních vztahů, až po digitalizaci produkce se uplatňují nové přístupy

a technologie v souladu s principy I4.0. Díky této masivní digitalizaci se také objevují na trhu nové obchodní modely díky čemuž dochází k propojení všech faktorů pomocí inovativních komunikačních sítí. [20]

Právě tyto sítě, jako jsou například Internet věcí (IoT), Internet služeb (IoS) či Internet lidí (IoP), jsou udávači možností I4.0. Neboť komunikační prostředky propojují jednotlivé entity a umožňují decentralizovaný přístup k celému firemnímu procesu. [3]

Všechny komponenty, které se v rámci I4.0 vyvíjejí nebo jsou již implementovány obecně splňují potřebné principy průmyslu 4.0. Základní počet těchto principů je 6 a jsou to: Interoperabilita, Virtualizace, Decentralizace, Práce v reálném čase, Orientace na služby, Modularita a rekonfigurabilita systémů. [3]

Využívané nástroje v rámci I4.0

Jak již bylo zmíněno v prostředí I4.0 se implementují či vyvíjejí nástroje splňující principy I4.0. Od hardwarových objektů přes softwarové algoritmy a programy až po funkční principiální modely.

U hardwarových objektů se hovoří především o robotizaci. Což znamená začleňování robotů do masivní průmyslové produkce v nových doposud neautomatizovaných aktivitách. Jako jsou například inventarizace ve velkoskladech, udržování bezpečnosti hlídaného parametru či manipulace s materiálem při jeho toku výrobním procesem.

Softwarové nástroje jsou komplexní programové algoritmy, které jsou určené pro mnoho účelů. Zahrnuty by zde mohli být nástroje MES (Manufacturing Execution System) a ERP (Enterprise Resource Planning). Ale také menší celky jako jsou programy pro řízení toku dat řízené právě systémem MES, nebo cloudové služby a to jak interní či externí ke společnosti využívající tuto metodu sdílení dat. V posledních letech lze vidět velký rozmach AI technologií, které jsou postaveny na propracovaných algoritmech. Jako příklad lze uvést perceptronové sítě či algoritmy strojového učení. [1]

Principiální modely vzhledem k I4.0 jsou svou určitou částí "KNOW-HOW"každého podniku. Z velké části jde o kombinace a propojení jednotlivých částí I4.0 například v síti IoS. [6]

Práce s daty v prostředí I4.0

V přístupu k průmyslovým datům a to nejen v I4.0 je na místě určit několik základních otázek týkající se žádaných dat. Tyto otázky jsou:

- Jaký je zdroj dat?
- Jak velký objem dat budu moci získávat a kolik ve skutečnosti potřebuji?

- Jak rychle je potřeba tyto data přenést či vyhodnotit?
- Jakou technologii uplatním pro jejich vyhodnocení?
- Po jak dlouhou dobu budu muset tyto data uchovávat?
- Vztahuje se na tyto data nějaká zákonná povinnost?
- Jakou formou získaná data budou zabezpečena?

Všechny tyto otázky budou muset být řešeny i v případě návrhu AAS. [20] [5]

U zpracovávání průmyslových dat v systémech I4.0 se nejčastěji hovoří o pojmu velká data. Jedná se o ohromné množství dat, které je možné získat pomocí senzorových sítí v rámci IoT. K těmto datům se dá přistupovat následujícím přístupem: Nastavení strategie velkých dat -> Identifikace zdroje big dat -> Přístup a uložení dat -> Analýza dat -> Vyvození výstupní informace. Jednotlivé body v tomto řetězci jsou často velké celky, které se zaobírají určitou specifikací. [4]

1.1.2 Vysvětlení pojmu AAS

Asset administration shell, doslovně přeloženo:"předmětová administrativní schránka", je, jak již bylo dříve zmíněno, jeden z novějších pojmů v prostředí I4.0. AAS ve své podstatě virtuálně reprezentuje námi zvolený objekt a zároveň zprostředkovává veškeré aktivity objektu v návaznosti na interakci s jeho prostředím v decentralizovaném průmyslovém systému. [7]

Při vysvětlování pojmu AAS lze začít vysvětlením významů slov v názvu Asset administration shell:

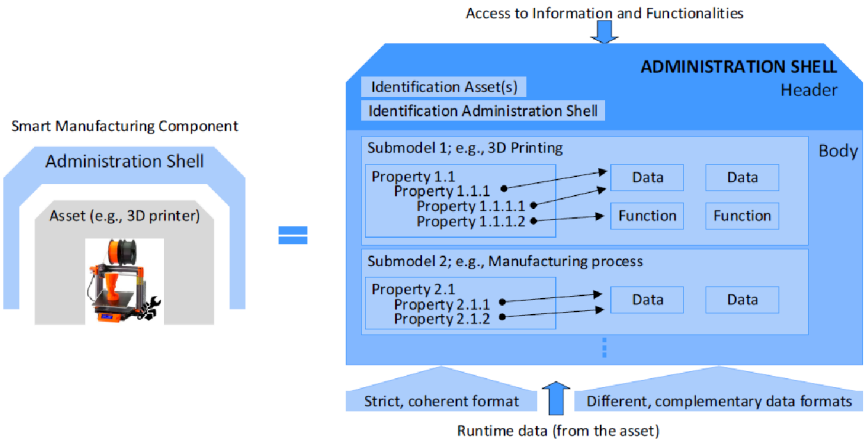
- Asset (předmět/objekt): je věc zájmu celého AAS, neboli objekt, kterému jsou přidány vlastnosti a data uchované v AAS, a s kterým prostřednictvím AAS komunikují ostatní komponenty
- administration (administrace): označuje aktivní a pasivní schopnosti AAS. Tyto pojmy budou rozebrány v následujících částech práce.
- shell (schránka): reprezentuje zapouzdření celého AAS do jednoho flexibilního, robustního celku.[12]

AAS jsou možné přisuzovat následující vlastnosti vzhledem k decentralizovanému systému a objektu, který zaštiťuje.

- Zajišťování komunikace: zpracovávání příchozích zpráv a odesílání zpráv v souladu s určeným komunikačním protokolem (např. příchozí zprávy obsahující žádost akce objektu, žádosti zaslání dat a odesílaná data v podobě žádaných dat či zadávání žádosti o provedení akce jiného AAS).
- Správa dat: v tuto část AAS lze ještě rozdělit na Interní data objektu, Statistická data, a ostatní data.
  - interní data objektu: jsou data o přímých vlastnostech objektu a jeho funkcích. Tato datová schránka vytváří specifičnost každého AAS.

  - statistická data: data, jež jsou získávána při vykonávání funkce objektu a jsou uchovávána pro další zpracování například pro vyhodnocování efektivnosti činnosti či odhadu stavu životnosti objektu.
  - ostatní data: zde jsou obsažena data, která nespadají do předchozích kategoriích. Například data získaná čistě pro zaslání dalšímu objektu na základě jeho žádosti.
- Zadávání příkazů k akci: AAS zadává podrobné příkazy objektu z databáze příkazů určené přímo tomuto objektu. Tyto akce jsou aktivovány žádostí o provedení dané akce jiným AAS, nebo jinou aktivní funkcí objektu. [2]

Strukturu mezi objektem a AS lze dobře vidět na následujícím obrázku viz. 1.1. AS jako takové obsahuje dvě rozčlenění na hlavička ("Header") a Tělo ("Body"). hlavička obsahuje unikátní identifikátor objektu obsaženém v rámci AAS a ID identifikátor samotného AAS. V tělu oproti tomu najdeme velké množství submodelů pro jednotlivé datové struktury a data samotná. [7]



Obr. 1.1: Struktura AAS [7]

1.1.3 Historie AAS

AAS se jako komponenta I4.0 poprvé představil roku 2015 v referenčním architektonickém modelu pro průmysl 4.0. Koncept samotného AAS byl nadále rozpracováván v roce 2016. V roce 2017 byl řešen kompozitní aspekt AAS.

Po sladění RAMI4.0 modelu s internetovou referenční architekturou IIC (Industrial IoT Reference Architecture) v roce 2017, platformou Průmyslu 4.0 a IIC, byl identifikovýn AAS jako jedinečný pro RAMI 4.0. Téhož roku se ukázal fakt, že koncept digitálního dvojčete a AAS se uchylují stejným směrem vývoje. Proto někteří popisují AAS jako implementaci DT (Digital Twin) v I4.0. [17]

V letech 2018 až 2020 byla zveřejněna a vylepšena řada specifikací pro koncept AAS. To umožňuje interoperabilitu v hodnotovém toku aktiva a umožňuje modularitu aktiva bez nutnosti výměny informačního modelu a API DT (Application Programming Interface DT). Tyto specifikace nadále pomáhají vývojářům nejen s implementací interoperabilního DT.

V prosinci 2019 bylo v rámci TC65(Technical Committee) zveřejněna skupina IEC WG24 „Asset Administration Shell for Industrial Applications“ a také ISO CD 23247 popisuje AAS jako součást svého výrobního rámce pro digitální dvojče. [17]

1.1.4 AAS v RAMI 4.0 modelu

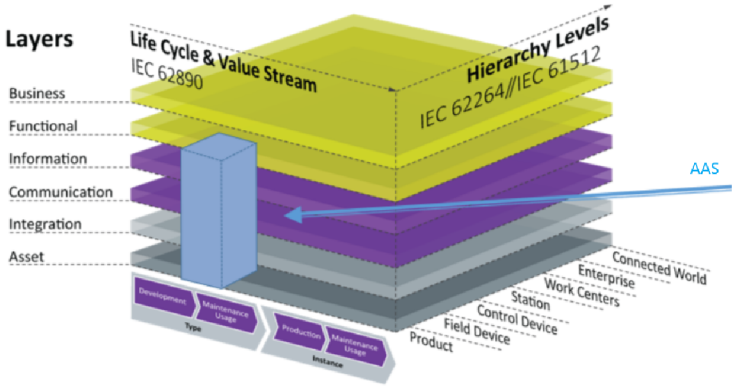
Jako jeden z mnoha pohledů na AAS lze zařadit pohled z RAMI 4.0 modelu, jako pohled nahlížející na AAS v širší spojitosti a kontextu vůči ostatním prvkůmv v I4.0. I zde se ovšem v odborné komunitě najdou odlišné názory na problematiku. Jako příkaldy uvádíme pohled od Standardizaton council Industrie 4.0 (SCI 4.0) a pohled společnosti SAP.

RAMI 4.0 model s AAS od SCI 4.0

SCI 4.0 má cíl iniciovat digitální produkční standardy v Německu a koordinovat je na národní i mezinárodní úrovni. Iniciativa urychluje normalizační procesy a organizuje a formuje nejen německý standardizační plán Industrie 4.0. SCI 4.0 také definuje potřebu nových projektů a soustředí se na jejich organizaci mezinárodních implementací. Jejich pohled na zasazení AAS do RAMI 4.0 modelu jde vidět na následujícím obrázku viz. 1.2 [8]

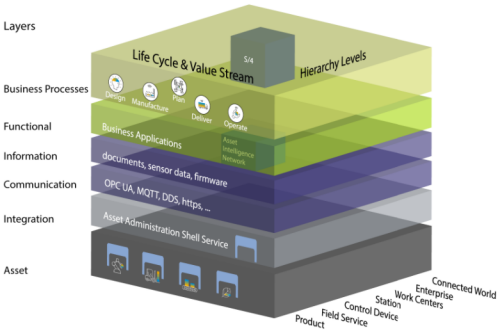
RAMI 4.0 model s AAS od SAP

Oproti tomu v Německé softwarové společnosti SAP se v přístupu k AAS objevuje spíše umírněnější postoj. SAP tvrdí, že zapojení AAS do výrobního procesu by neměl narušit již stávající zavedený systém, a AAS by měl být spíše doplňkem v I4.0.



Obr. 1.2: AAS v RAMI 4.0 modelu SCI 4.0 [8]

Tento postoj je znázorněn v modelu RAMI 4.0 viz. obrázek 1.3, kde lze vidět, že oproti modelu od SCI 4.0 je AAS zakomponováno pouze ve spodních vrstvách implementace. [9]



Obr. 1.3: AAS v RAMI 4.0 modelu od SAP [9]

1.2 Přístup k AAS v I4.0

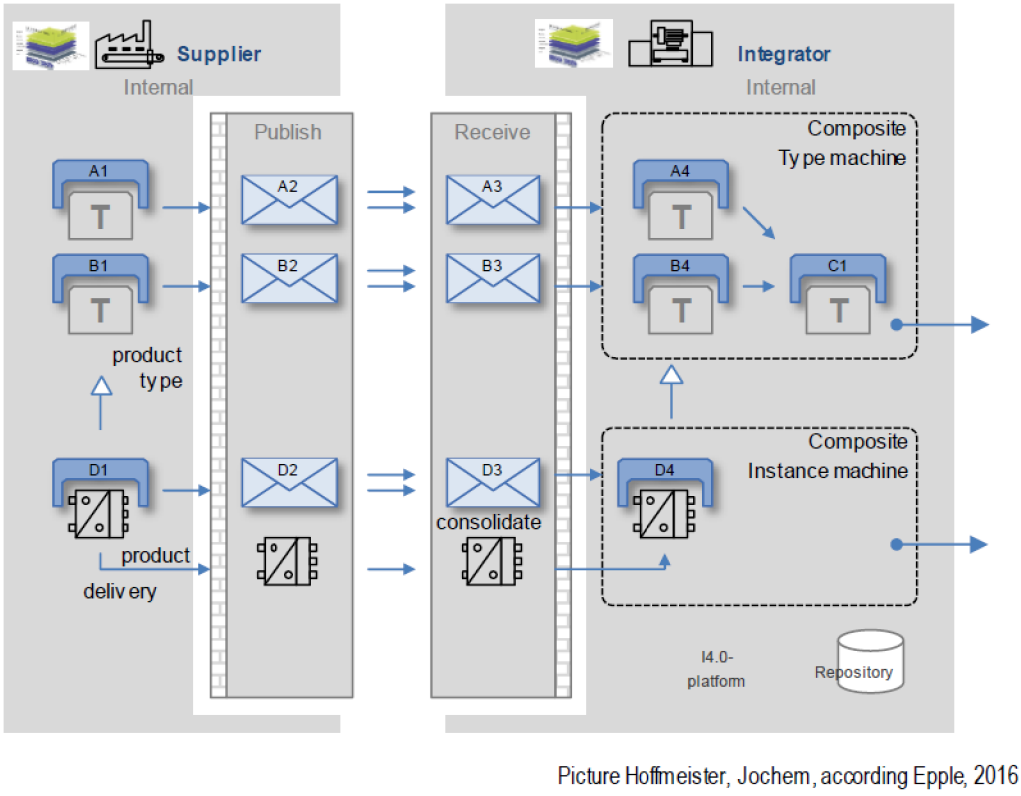
V této části práce se představí princip manipulase s AAS napříč podniky, a také několik pohledů na AAS z I4.0.

1.2.1 Manipulace s AAS napříč podniky

Ve výměně AAS mezi jednotlivými partnery, nejčastěji dodavatelem a příjemcem, se AAS přenáší v uzavřených datových kontejnerech obsahující samotné AAS připravené na implementaci, podpůrnou dokumentaci a ostatní potřebné či dohodnuté dokumenty ze strany příjemce. U těchto datových kontejnerů musí být zvolena adekvátní ochrana dat jako celku, dodržovaná všemi stranami, které s těmito datovými

kontejnery přijdu do styku. Po převzetí a extrahování dat z přenosového kontejneru je příjemce dodržovat pravidla o zacházení s těmito daty dohodnuté mezi oběma stranami. [10]

Takovouto výměnu dat lze vidět na následujícím ilustračním obrázku viz. 1.4



Obr. 1.4: Tok dat AAS mezi dvěma partnery [10]

1.2.2 Integrace AAS do průmyslového procesu I4.0

Při integraci AAS do automatizovaného systému I4.0 se musí na samotné AAS myslet ihned od začátku implementace všech zařízení zahrnutých do zvoleného procesu. AAS je nástroj vhodný pro maximálně decentralizovaný průmyslový systém. AAS lze přiřadit jakémukoliv HW i SW prvku a právě díky AAS získá takový to prvek určitou formu autonomie což je v decentralizovaném systému žádané.

Jak již bylo zmíněno, pokud je chtěné zavést v budoucí decentralizované síti prvek AAS, musíse na každý prvek od počátku jeho vývoje pohlížet od základu formou AAS. AAS se následně stane základním kamenem sítě, a proto se musí klást velký důraz na prvotní univerzálnost a funkčnost metamodelu celého AAS. Z takového modelu se následně vychází u tvoření specifických AAS pro každý prvek zapojený nejen ve výrobním procesu. [2]

AAS a již vyvinuté prvky

Samozřejmě jde AAS aplikovat i na již vyvinuté a odzkoušené prvky, ale ztrácí se zde ovšem benefit v podobě vedení evidence o detailech např. v konstručních prvcích s určitou dobou životnosti. Pokud monitorujeme zařízení od jeho vývoje, je mnohem jednoduší sestavení a využití jeho DT v AAS. Je potřeba podotknout fakt, že někteří odborníci se neshodnou, zda AAS a DT není ten samý prvek I4.0 s jiným úhlem pohledu. V tomto dokumentu je přikláněno k rozdělení AAS a DT na dva odlišné prvky, a to především z důvodu, že DT neplní aktivní administrativní funkci v takovém rozsahu jaký poskytuje AAS. [13]

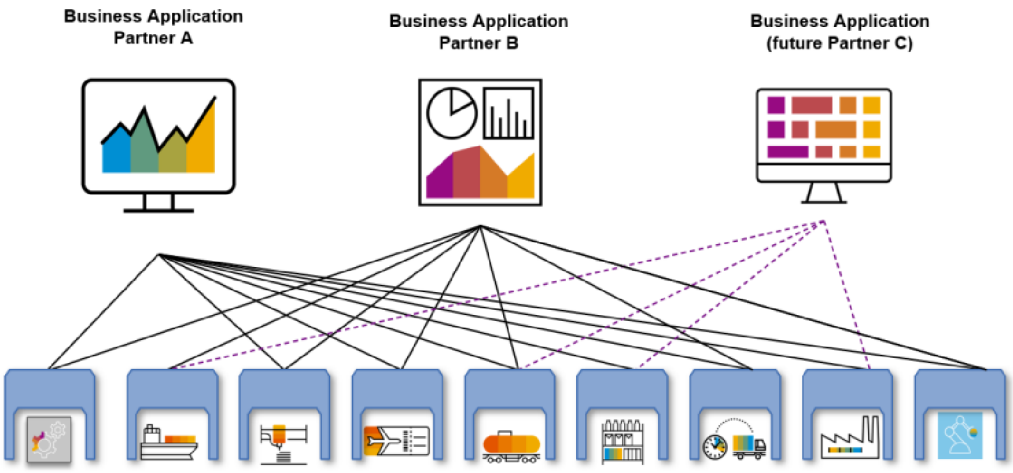
1.2.3 Open Industry 4.0 Alliance a AAS

Na hannoverském veletrhu v roce 2019, se sedm společností z oblasti strojírenství, průmyslové automatizace a softwaru rozhodlo založit alianci Open Industry 4.0 Alliance. Tato aliance je otevřená všem společnostem jejíž zájmem je rozšíření digitalizace a implementace I4.0 v nejen v EU. Toto uskupení nemá za účel vyvíjet nové systémy či standardy, ale pouze zpřístupnit více společnostem již standardizované systémy pro I4.0 a tím dosáhnout rychlejší a širší uplatnění nových metod v rámci I4.0. Jejich pohled na AAS je takový, že každá společnost bude mít určitou možnost přistupovat ke Cloudu, kde bude mít již předvytvořené metody a postupy pro práci s AAS a AAS samotné. [11]

Tímto přístupem se předpokládá dosažení obchodních aplikací budoucnosti, které budou brát v potaz kmenová data aktiv a budou mít přístup k AAS jako k tzv.“single source of truth” (Volně přeloženo: „jediný zdroj pravdy“). Tento princip je zobrazen ilustrativně na obrázku. viz.1.5 [11]

1.2.4 Shrnutí integrace AAS do průmyslového procesu I4.0

Obecně tedy lze říci, že decentralizované systémy postavené na systémech AAS jsou velmi robustní, flexibilní a lehce modulovatelné. Oproti tomu s AAS nelze zanedbat větší složitost na vytváření takového systému s vyššími prvotními náklady na vytvoření celkového výrobního procesu. I přes tyto nevýhody se v dlouhodobém časového horizontu implementace AAS do průmyslového procesu I4.0 nejspíše vyplatí.



Obr. 1.5: Open Industry 4.0 Alliance a AAS obchodní aplikace budoucnosti s AAS [11]

1.3 Životní cyklus objektu s AAS

Jak již bylo zmíněno, AAS se v ideálním případě vytváří již při vývoji samotného objektu. Tudíž při vývoji zařízení se dá uvažovat starší přístup z průmyslu 3.0, či z pohledu I4.0. Tyto dva postupy se od sebe liší především v tom, že vývoj z pohledu I4.0 klade velký důraz na softwarovou část vývoje, což znamená, že vše co lze digitalizovat a simulovat se provede ve virtuálním prostředí. Tímto postupem se dá předcházet nechtěným defektům nově vyvinutého zařízení a tím ušetřit za výrobu případných prototypů. Také to sebou nese výhodu v získávání užitečných statistických dat k zařízení, které následně lze využít v AAS například formou DT. [2]

1.4 Pasivní část AAS

Pasivní část AAS lze definovat jako veškerá virtuální data obsažená v AAS.

AAS vždy musí obsahovat určitý objem dat, a to statická, nebo také dynamická data. Statické data jako například základní parametry, statické údaje, nebo také metamodely a submodely jednotlivých prvků v AAS. Dynamicky se měnící data, mohou například obsahovat měřené fyzikální veličiny z objektu AS v reálném čase či vnitřní proměnné. K těmto typům dat následně může mít přístup aktivní část AAS, která s nimi různými způsoby interaguje.

1.4.1 Type a Instance objektu v AAS

Každý objekt v AAS vychází z prvkového typu TYPE (označní assetkind: type) a prvkového typu INSTANCE (označní assetkind: instance). Oba tyto prvky vychází z jednoho metamodelu. Metamodel určí strukturu a data jako základ pro metamodel typu TYPE, ze kterého bude vycházet TYPE s již danými parametry, které se přiřadí k určitému druhu objektu. Z typu TYPE se následně určuje metamodel INSTANCE již přímo jednotlivým existujícím instancím prvků objektů AS. [10]

Obecně se v časové ose typy objektu AS dají představit asi takto:

- Type
  - vývoj – Definuje se funkčnost a princip existence prvku
  - použití – Vytvoření důležitých a nutných parametrů, které musí každý prvek obsahovat. Deklarace jeho možných typů vlastností a množství datových prvků. [10]
- Instance
  - produkce – Převzetí "šablony"prvkového typu Type a specifikování jednotlivých prvků požadovaných pro tento prvek.
  - použití – Využívání tohoto prvku např. interakcí s metodami pracující v oblasti tohoto SW prvku. [10]

V časové ose takovéhoto prvku se nachází čtyři druhy vazeb. Nejdůležitější je ta mezi samotným typem Type a typem Instance. Tato vazba zaručí možnost úpravy všech Instancí typu Type, jedinou změnou právě v tomto typu Type. Jako druhá nejdůležitější je zpětná vazba mezi prvkovými typy.[10]

1.4.2 Metamodel AAS

V této části budou představeny hlavní části metamodelů AAS. Jako jedeny z nejdůležitějších prvků v AAS jsou informace o samotném objektu v rámci AS (AssetInformation) s jejich submodely.

Jak již bylo zmíněno každý objekt zaštítěný v AAS má typ Type a Instanci, Toto rozlišení se provádí právě v parametru "AssetInformation/assetkind". [10]

Z důvodu možnosti, že typ Instance, občas i Type, vychází z nadřazeného prvku, existuje parametr odvozeno z (derivedFrom). Zde je obsažen odkaz právě na zmíněný nadřazený prvek. V důležitých infomracích o AAS, je také nutné mít obsažený unikatní identifikátor AAS a objektu samotného. [10]

Submodely, které jsou částí metamodelu AAS, se dají členit do skupin, jako jsou například: vlastnosti, operace, datové kolekce, atd... . Každý takový to submodel musí vycházet ze sémantické definice. Na takovou to definici může odkazovat buď to přímo, nebo může odkazovat na popis konceptu submodelů, kde se dá dohledat definice určená právě tomuto submodelu. Každá taková definice by měla být odvozena

od již existujících standardů (např. ECLASS, IEC DDD). Tento přístup zajišťuje interoperabilitu AAS. [10]

Metamodel se obecně rozlišují mezi prvky, které jsou identifikovatelné, odkazovatelné, nebo ani jedno.

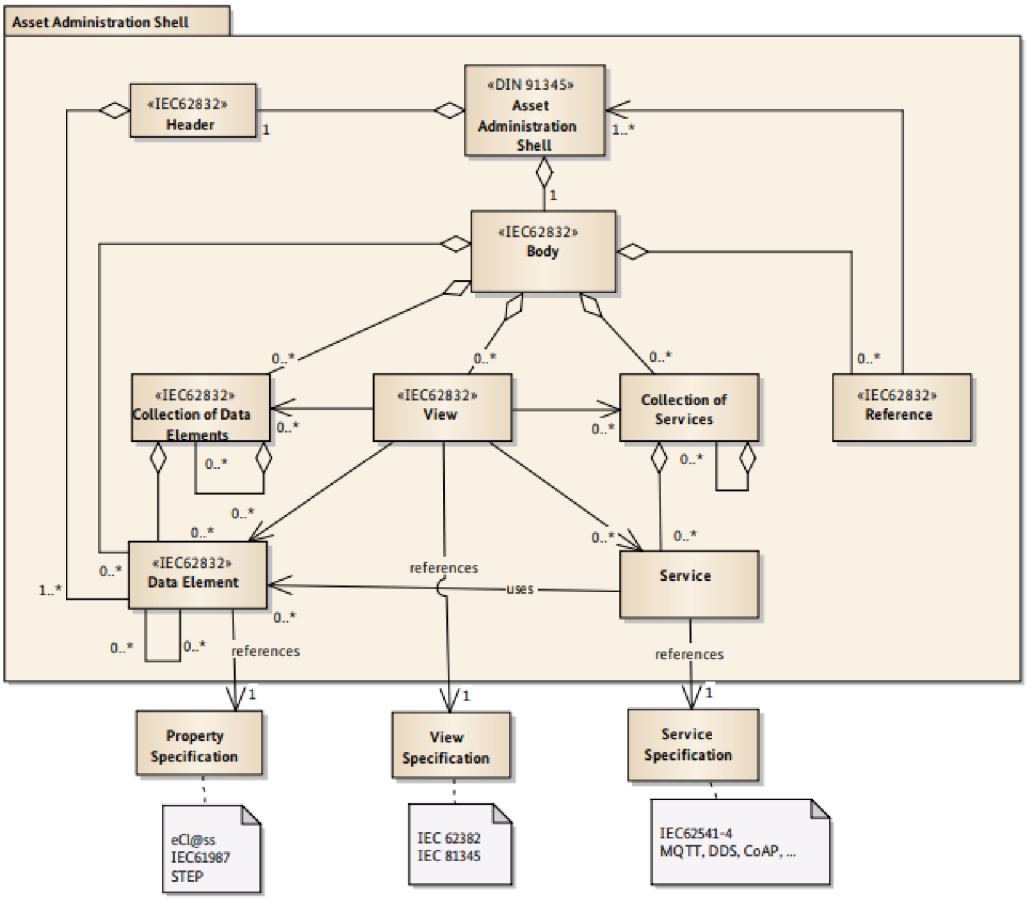
V případě nutnosti vytvoření atributů nad rámec metamodelu, je na místě využít šablon určující formu specifických dat (často se volí šablona odpovídající standardu IEC 61360).

Další informace o submodelech AAS a informací s nimi spojenými lze najít v práci ZVEI viz. [10]. Z důvodu rozsáhlosti této problematiky by se tato část obsahově nevešla do této práce.

1.4.3 Vnitřní struktura AAS

Zde bude představena struktura AAS v oblasti vnější vrstvy. Struktura se skládá z Headru a Body a několika dalšími na sebe navazujícími prvky. Každý prvek v této struktuře lze rozepsat na další podvrstvě do větších detailů. Struktura je obecná, a tudíž ji lze s uživatelskou implementací realizovat dle specifických požadavků systému, do kterého se AAS implementuje. Kořenový uzel modelu je samotné AAS stanovené dle normy DIN 91345. Vizualizaci této struktury lze vidět v diagramu na obrázku 1.6 [14]

- záhlaví (Header): záhlaví obsahuje minimální počet informací veřejně dostupných, za účelem identifikace AAS a objektu samotného. Dle německého ZVEI by se zde také měl nacházet posouzení o zabezpečení odkazu na tělo(body) AAS. Také by zde měl být umožněn případný přístup díky identifikátoru k typu Type a Instance. Identifikátor může být získáván přímo z AAS či odkazem na úložiště, kde se daný identifikátor AAS nalézá.[14]
  Mezi základní prvky, které musí být v záhlaví obsaženy, jsou: globální unikátní id (identifikátor), informace o verzi administrace a odkaz na další informace o identifikátoru objektu.[12]
- tělo (body): Tato část se obecně uvažuje za pasivní kontejner dle normy IEC 62832 (popisující digitální továrnu), obsahující data s vlastnostmi a podpůrnými pohledy. U definice těla v AAS se přidávají další prvky a to služby a reference s možností vyvolání určitých služeb. Tímto se tělo stává aktivním prvkem a ne pouze pasivním datovým kontejnerem.[14]
- Kolekce datových prvků (Collection of Data Elements): Tento člen má napomoci s prací s datovými prvky tím, že jsou zde předem definované, ideálně standardizované, struktury dat. Je zde i jistá podoba se souborovými systémy operačních systémů pro obecné použití, a tudíž jsou také označovány za částečné modely. U datových struktur se lze setkat s normami eCl@ss, ISO 10303



Obr. 1.6: diagram struktury AAS [14]

(pod formálním názvem "STEP"), IEC 61897 a IEC 62832.[14]

- datový prvek (Data Element): Datový prvek je pár dvou parametrů a to klíče a hodnoty, vztahující se ke statické či dynamicky se měnící hodnotě objektu. Klíč zde představuje odkaz na celosvětově uznávanou sémantiku datového prvku. Hodnota je konkrétní údaj v daném datovém typu, vztažený či získaný právě k objektu AS. Měly by se zde vyskytovat pouze standardizované datové prvky například standardem IEC 61360. Modelování vlastních datových prvků například, pro využití ve funkci objektu, se realizují v jiné části AAS.[14]
- pohled (View): Tato třída realizuje filtr obsahu AAS. Vzhledem k tomu, že AAS může obsahovat stovky datových prvků, je na místě uvažovat o zpřístupnění pouze relevantních dat určitým uživatelům či nástrojům. Každý prvek, který bude komunikovat s AAS bude mít přiřazený pohled do AAS ze seznamu pohledů, který mu umožní přístup k již zmíněným relevantním prvkům. Tento přístup zaručí nejen zabezpečení systému, ale také zvýšení efektivnosti a přehlednosti přístupu a komunikaci mezi AAS a externím prvkem/uživatelem. [14]

Dle konceptu v IEC 61832 by seznam pohledů měl být na centrálním úložišti s odkazy na datové prvky. Tento přístup je ovšem u AAS velmi nevhodný nejen, že je to velmi centralizovaný prvek, ale také zamezuje uživatelsky modifikovatelné pohledy pro jednotlivé AAS.[14]

- kolekce služeb (Collection of Services): AAS může obsahovat, a v případě komplexního aktivního prvku, i musí obsahovat, kolekci možných volání služeb, které je objekt schopný vykonávat. Tato třída by v případě velké databáze funkcí měla umožňovat seskupovat spolu související funkce do jednotných skupin pro lepší přehlednost a efektivnost jejich volání. Tento prvek nepopisuje zatím žádný standard.[14]
- služba (Service): Třída služba poskytuje rozhraní pro spustitelné funkce AAS. Z velké míry se zde nachází odkazy na funkce vyšší úrovně specifického obejktu. Tyto funkce často volají vnitřní pod funkce s možnostmi další komunikace či administrativních úkonů, nebo sběru a práce s daty. Pokud má služba nastavitelné parametry vstupu a výstupu při jejím vykonávání, je nutné je definovat, pokud nejsou implicitně nastavené pro případ nemodifikovatelnosti žadatelem služby.

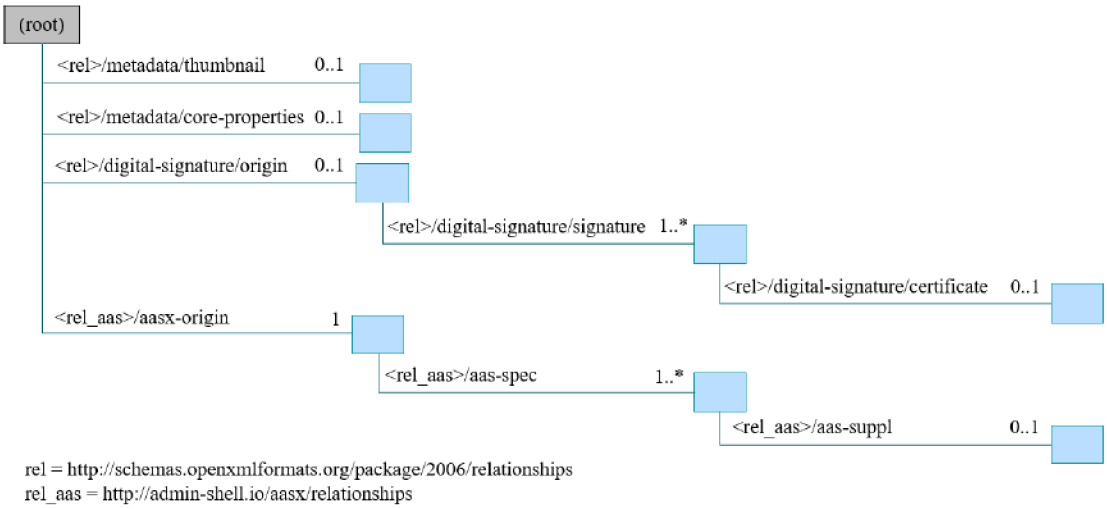
  Je zde hojně využíváno komunikačního schématu žádosti a odpovědi, kde žadatel služby podá žádost do AAS a čeká na jeho odpověď.

  Také lze zde najít služby modifikující modely a stavy samotného AAS, pokud k tomu bylo AAS navrženo. Tyto služby mohou inicializovat objekt, autorizovat uživatele AAS, pracovat s vnitřními parametry AAS obecně.

  Třída služba nespadají pod žádný standard. Je ovšem nutné říci, že například při využití komunikace OPC UA je využita norma IEC 62541-4 (OPC UA Services), která určité služby již definuje automaticky.[14]
- reference (References): AAS jako takové může k službám, které vykonává, dále využívat služeb dalších AAS. Proto se do AAS zanesla třída reference, která obsahuje informace o nejčastější interakci s ostatními AAS a také případné odvození aktuálního AAS od jeho předchůdců. Tyto parametry se mohou dynamicky měnit v rámci decentralizovaného systému, kterého je AAS součástí. Tyto informace napomáhají například, s plánováním co nejefektivnějším rozložením pozic a vazeb mezi AAS. [14]

1.4.4 Typ souboru obsahující data AAS

V mnoha případech bude dozajista potřeba manipulovat se samotným AAS jako celkem, či s jeho interními daty. Aby nebylo nutné zasahovat přímo do zdrojového kódu AAS a přehrávat jeho části, je nutné aby AAS mělo jednotný souborový formát, který bude zajišťovat jednotnost, kompaktnost a jednoduchý přístup jednotlivých



Obr. 1.7: Diagramová ukázka typů vztahů v souboru AASX [10]

částí AAS. [10]

Řešením je souborový formát AASX. Tento formát je navržen aby splňoval následující: obecnost formátu zahrnující AAS a všechy jeho data, možnost použití pro přesun mezi společnostmi, přístup k interním datům, právní neomezení, licenční nezávislost s ohledem na mezinárodní standardy, existence API pro vytváření, čtení a zápis, možnost šifrování a zásady autenticity.[10]

Formát AASX

Formát AASX byl odvozen od otevřené souborové konvence standardů, s převzetím jeho vlastností. I přes to AASX má definovano několik konvencí jako jsou například: splňování pravidel ISO/IEC 29500-2:2012, Příponu .aasx, ikonu AASX, atd... . [10]

Pro splnění výše uvedené normy je nutné definovat logický, fyzický a bezpečnostní model.

- logický model: Obrázek viz.1.7 zobrazuje příklad typů vztahů pomocí URI s odpovídajícími zdrojovými soubory. U instancí vztahů lze také najít unikátní ID s cílovým zdrojem.

Většina typů vztahů bylo definováno v rámci ISO/IEC 29500-2:2012 (jako například základní vlastnosti, digitální podpis, miniatury, atd...), a proto je netřeba znovu definovat.

Konvence souborových názvů

Díky využívání vztahů standardu ECMA-375 je umožněno vyhledávání souborů v balíčku AASX v nezávislosti na názvu souboru. Je tedy možné uložit soubor do

různých balíčků, ale protože všichni využívají stejný typ vztahů k cílení na tento soubor, vždy je umožněno tento soubor nalézt. [10]

Pro přehledný přístup k balíčkům se definovali konvence pojmenovávání souborů uvnitř AASX:

- /aasx/ je společná předpona pro všechny soubory s obsahem specifických pro balíček AASX.
- /aasx/aasx-origin je cílem vztahu aasx-origin neobsahující obsah.
- /aasx/data.<přípona> je cílem vztahu aasx-origin s příponou ve tvaru "xml"nebo "json"v závislosti na serializaci.

Šifrování souborů v rámci AASX

Norma ISO/IEC 29500-2:2012, ze které se vychází říká, že balíčky založené na ZIP nesmí obsahovat šifrování. Balíček otevřené souborové konvence je ale možný šifrovat jinými způsoby.

Šifrování lze využít například během výměny nebo použití DRM (Digital rights management) pro distribuci souboru AASX. Pokud je chtěné šifrování části či celé AASX je nutné se ujistit, že je použití šifrování vhodné. Důvěrná data, která mají být chráněna musí být obsaženy v souboru, pokud tomu tak není tak tato cesta není doporučována.

Pokud je potřeba soubor zabezpečit čistě na omezenou dobu, například pro přesun či odložení dat, je vhodnější k tomu využít externí aplikaci, která toto zabezpečení zprostředkuje. [10]

1.5 Aktivní část AAS

Aktivní část AAS lze definovat, jako tu část AAS, kde dochází k interakci s jinými prvky vně i uvnitř AAS. Patří sem komunikace s ostatními AAS (nebo jinými prvky), operace v rámci vnitřní struktury samotného AS, a interakce s objektem AS. Interakce tedy představují vykonávání funkcí a metod obsaženy v AAS. Operacemi v rámci vnitřní struktury se rozumí interakce s pasivní částí AAS.

1.5.1 Identifikace prvků

V rámci domény chytré továrny, jsou identifikátory velmi důležité a podstatné prvky, sloužící k jednoznačné identifikaci mnoha různých prvků, a proto jsou základním prvkem popisující AAS. [10]

Identifikátory jsou zejména potřebné pro:

- AAS
- objekt AS

- submodel instance a submodel šablony
- definice vlastností a popisů nacházející se v externích úložištích

Identifikace se využívá především ve dvou případech:

- k jednoznačnému rozlišení všech prvků v AAS
- k vytváření vazeb mezi prvky a jejich submodely, metamodely, atd...

Druhy identifikátorů

Globálně se identifikátory vyskytují především IRDI (International Registration Data Identifier), IRI (Internationalized Resource Identifier) nebo vlastní.

- IRDI => vychází z norem ISO 29002-5, ISO IEC 6523 a ISO IEC 11179-6. Jejich vytváření probíhá v rámci mezinárodní standardizace.
- IRI => nebo také URI a URL podle RFC 39862.
- vlastní => samozřejmě lze realizovat i vlastní typ identifikátorů. Tento přístup je ovšem vhodný pouze pro interní použití daných objektů, kterým je tento identifikátor přiřazen.

Pokud jsou použity URI či URL identifikátory je možné díky nim se odkazovat na standardizované datové struktury a služby. Díky tomu se získá poměrně univerzální přístup nejen v komunikaci v rámci komunikační sítě, ale také ulehčení hledání jednotlivých prvků a vytváření nových objektů. Mimo globálních identifikátorů existují také identifikátory, které jsou jedinečné pouze v rámci definovaného jmenného prostoru. [10]

AAS do značné míry podporuje globální identifikátory. Jsou ovšem případy kdy využití globálního identifikátoru je neefektivní. Proto se často interním třídám submodelům a metamodelům v AAS, přiřazují identifikátory idShort. Tyto identifikátory dědí vlastnosti z abstraktní třídy reference, čímž se na tyto závislé prvky odkazují. Tento přístup ovšem vyžaduje speciální adresovací funkci pro zápis i čtení identifikátorů. [10]

1.5.2 Události AAS

Událostí se rozumí veškerá akce AAS mimo běžný pracovní režim. Události se přenáší pomocí zpráv, které obsahují dokumentaci o události a popřípadě samotná data, týkající se vykonávání události. [10]

Události mohou být dopředné, ale i reverzní.

- Dopředná událost je interakcí s AAS od jeho poskytovatele k integrátorovi (uživatel AAS). Může se jednat například o aktualizace firmwaru, profesionální údržbu případně nastavení nového produktu.

- Reversní událost naopak vede od integrátora (uživatele AAS) směrem k jeho poskytovateli. Může se jednat o diagnostické hlášení o dlouhodobém chodu AAS, či hlášení o chybě, apod. [10]

U samotného AAS se rozlišují události dle směru toku dat. Jedná se o data určená pro příjem AAS, či k exportu dat události od AAS. Zprávy určené k exportu se přenášejí pomocí sítě, která je již implementovaná v AAS (OPC UA, MQTT, atd..) do dalšího AAS, případně až k exportu mimo síť. U vstupní události je postup stejný pouze s inverzním směrem toku dat. [10]

Typy událostí mohou být: strukturální změny AAS, vylepšení atributů AAS, data o provozu AAS, monitorování stavu AAS, infrastrukturální změny, změny v úložišti AAS, bezpečnostní události, alarmy a ostatní události. [10]

Jak již bylo zmíněno, každá událost se přenáší pomocí jednotně formátované zprávy. Taková zpráva by měla obsahovat následující atributy: zdroj události, sémantické ID zdroje, pozorovatelný odkaz, sémantické ID odkazu, téma události, předmět události, časová známka události, zatížení události. [10]

1.5.3 Komunikace u AAS

Při kompletním pohledu na AAS se lze setkávat s dvěma druhy komunikace viz. 1.6. Jedná se o komunikaci mezi fyzickým objektem a virtuálním AS a o komunikaci napříč sítí složené z vícero prvků realizovaných pomocí AAS. [12]

Komunikace fyzického objektu s virtuálním AS

Zde se často vychází z dostupného komunikačního rozhraní a protokolu, které je umožněné realizovat se samotným objektem. Z důvodu neomezenosti výběru objektu je tato komunikace opravdu široký pojem. V případě, že zařízení kterému je AS určeno, není schopno realizovat na vlastní HW platformě virtuální část AAS, je nutné vytvořit komunikační propojení mezi objektem a HW, které obsahuje AS, či k němu má přímý přístup. Může se jednat například o komunikace: I/O, PROFIBUS, CAN, RS 232, RS 485, Ethernet/IP a mnoho dalších. [5]

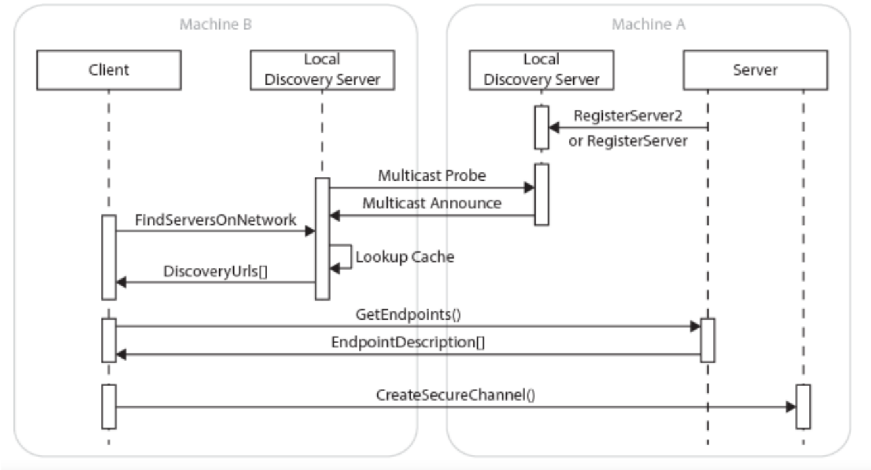
Tento přístup se musí volit například pokud objekt v AAS představuje samotný senzor (nebo jejich skupina), inteligentní kompaktní zařízení, embedded systém, nebo například člověk, u kterého se musí volit specifický komunikační přístup. [5]

V každém případě jsou tyto komunikace prováděny interně v AAS a nemá k nim tudíž žádný jiný objekt přímý přístup.[7]

Komunikace mezi jednotlivými AAS

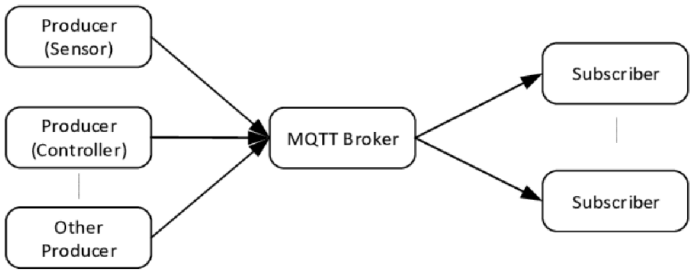
Tato komunikace je nejčastěji realizována komunikací pomocí OPC UA nebo MQTT v rámci interní podnikové sítě. Přičemž nejčastější využívaný protokol je TCP/IP. Každý z přístupů řešení komunikace má své výhody i nevýhody. [15]

- OPC UA: je velmi rozšířený komunikační architektura pro průmyslové využití. Funguje na principu server - client, tudíž nepotřebuje žádný centrální prvek. V sítích o velkém počtu prvků se ovšem začne projevovat značná latence zpráv, a proto se přidávají další asistenční prvky této sítě, jako je místní vyhledávací server (local discovery server: LDS) nebo více zprávové rozšíření (multicast extention: ME). LDS je prvek zajišťující efektivnější propojování jednotlivých prvků mezi sebou. Prvek ME zajišťuje funkci rozeslání zprávy vícero prvkům v komunikační síti najednou. Příklad komunikačního diagramu lze vidět na obrázku viz. 1.8 [15] [7]



Obr. 1.8: Ukázka diagramu OPC UA s LDS a ME [15]

- MQTT: není sice tak rozšířený jako OPC UA, ale to nemá vliv na jeho efektivitu. MQTT na rozdíl od OPC UA je s centrálním prvkem, nachází se zde broker, který řídí a zprostředkovává komunikaci mezi jednotlivými zařízeními. Celý princip funguje na bázi zveřejnění a odběru dat. Každé zařízení může zveřejnit data a každé zařízení může data obdržet. Tento princip je znázorněn na obrázku viz. 1.9 [16] Pro zaručení spolehlivosti sítě s MQTT a pro přiblížení se decentralizované bázi komunikace lze přidat neurčitý počet redundantních brokerů. MQTT je vhodnější pro sítě o méně prvcích, neboť dosahuje kratší latenční doby.

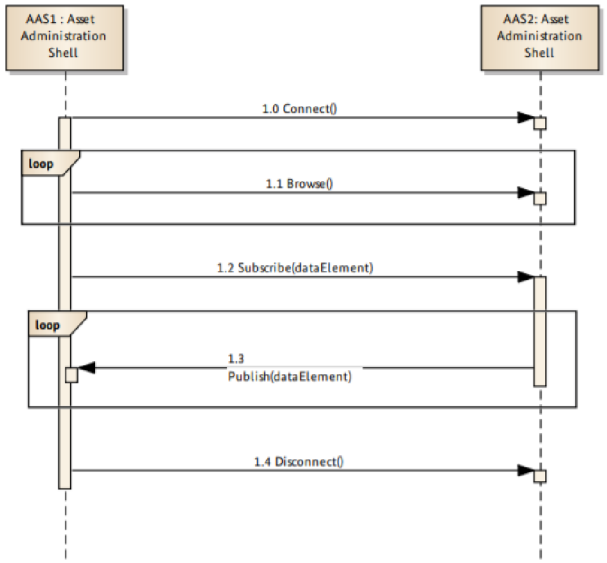


Obr. 1.9: Ukázka MQTT architektury [15]

Diagram komunikace mezi jednotlivými AAS

Na následujícím obrázku viz.1.10 je znázorněná komunikační interakce mezi dvěma AAS v režimu klient/server.

AAS1 naváže komunikaci s AAS2. Následně si AAS1 prochází dostupné datové prvky a služby, které AAS2 poskytuje. Jako další krok zažádá o akci zaslání datového prvku. AAS2 odešle datový prvek AAS1. AAS2 v poslední fázi komunikaci ukončí.[14]



Obr. 1.10: Diagram principu komunikace mezi AAS [14]

Tok dat ve výrobním procesu

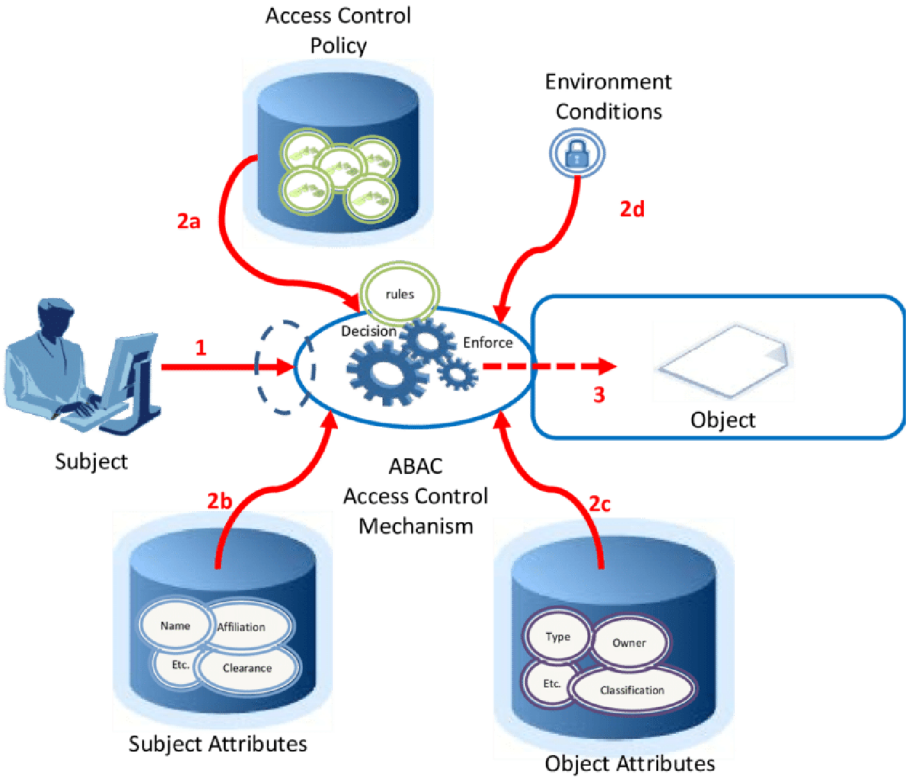
Ve firmě využívající AAS je v ideálním stavu pouze zadán cíl výroby manažery podniku a výroba se přizpůsobí automaticky. Z pohledu řízení výroby docházíme k rozdělení funkce AAS do 3 skupin.[7]

- žadatel o službu (service requester: SR): Nejčastěji se jedná o produkty, které hledají SP pro dosažení požadovaného cíle, a tím vlastně řídí vlastní výrobu na základě určených parametrů.
- poskytovatel služby (service provider: SP): Jedná se nejčastěji o prvek interagující s produktem či daty týkající se produkce jednotlivých produktů či produktů samotných. Je důležité, aby každý SP prvek mohl nabízet své služby SR, jinak by nikdy nebyl začleněn do výrobního procesu.
- společná část: Zde jsou zahrnuty jednotné komunikační operace, protokolování, strukturovaný přístup k datům.[7]

1.5.4 Zabezpečení AAS

Protože je AAS hlavním prvkem manipulující s objektem AS a hlavním přístupovým prostředím k jeho datům, je nutné vzít v potaz i bezpečnostní otázku AAS.

Bezpečnost AAS se řeší především z pohledu výměny dat mezi jednotlivými AAS, v rámci interní průmyslové sítě např. IoS. Tyto interní sítě jsou považovány za bezpečné a proto se nepředpokládá příjem dat do AAS obsahující poškozující, či jinak nepřátelský kód od odesílatele zprávy.[10]



Obr. 1.11: Základní scénář ABAC bezpečnostního přístupu [18]

U AAS se využívá především bezpečnostní přístup ABAC. Tento přístup je vhodný především díky kontrolování bezpečnostního statusu na základě zpracovávání určených atributů obou účastníků komunikace. Tento proces znázorňuje obrázek 1.11 základní scénář ABAC bezpečnostního přístupu. [18]

Definice vnitřních operací AAS, které patří do aktivní části AAS, lze nalézt v práci: Details of the Asset Administration Shell. Part 2 -Interoperability at Runtime - Exchanging Information via Application Programming Interfaces (Version 1.0RC01). [19]

2 Praktická realizace AAS pro operátora

V teoretické části byl proveden rozbor pojmu AAS, jenž dává základ návrhu a celkové implementaci AAS pro Operátora jemuž se věnuje následující část.

2.1 Návrh přístupu ke konceptu AAS

Při koncepčním návrhu AAS muselo být bráno v potaz jeho budoucí využití v testbed systému robotického barmana nacházející se na VUT FEKT UAMT.

Výsledná realizace byla také zaměřena na možnost implementace na jiných průmyslových platformách, a z tohoto důvodu byl návrh proveden obecně s prvky nevyužitými v konkrétní implementaci na platformě pro operátora.

Také byla snaha o vytvoření standardu pro AAS. Na tomto standardu byla prováděna spolupráce s další bakalářskou prací zabývající se implementací AAS na platformě PLC. Tento fakt byl také zapříčiněn tím, že obě implementace, AAS pro operátora i AAS pro PLC, by měly být navzájem kompatibilní.

2.1.1 Obecný přístup a účel AAS

Vývoj AAS byl zaměřen pro specifické použití v podobě operátora. Při jeho vývoji bylo nutné uvažovat o základních principech využitelných na vícero zařízeních.

Jako základní vlastnosti, které by AAS v implementovaném systému mělo obsahovat, jsou například: možnost volání služby od AAS, přístup k pasivním datům týkající se AAS a možnost část dat editovat.

Při vývoji museli být tyto vlastnosti implementovány s ohledem schopnosti implementace daných postupů a operací i na jiných SW a HW platformách.

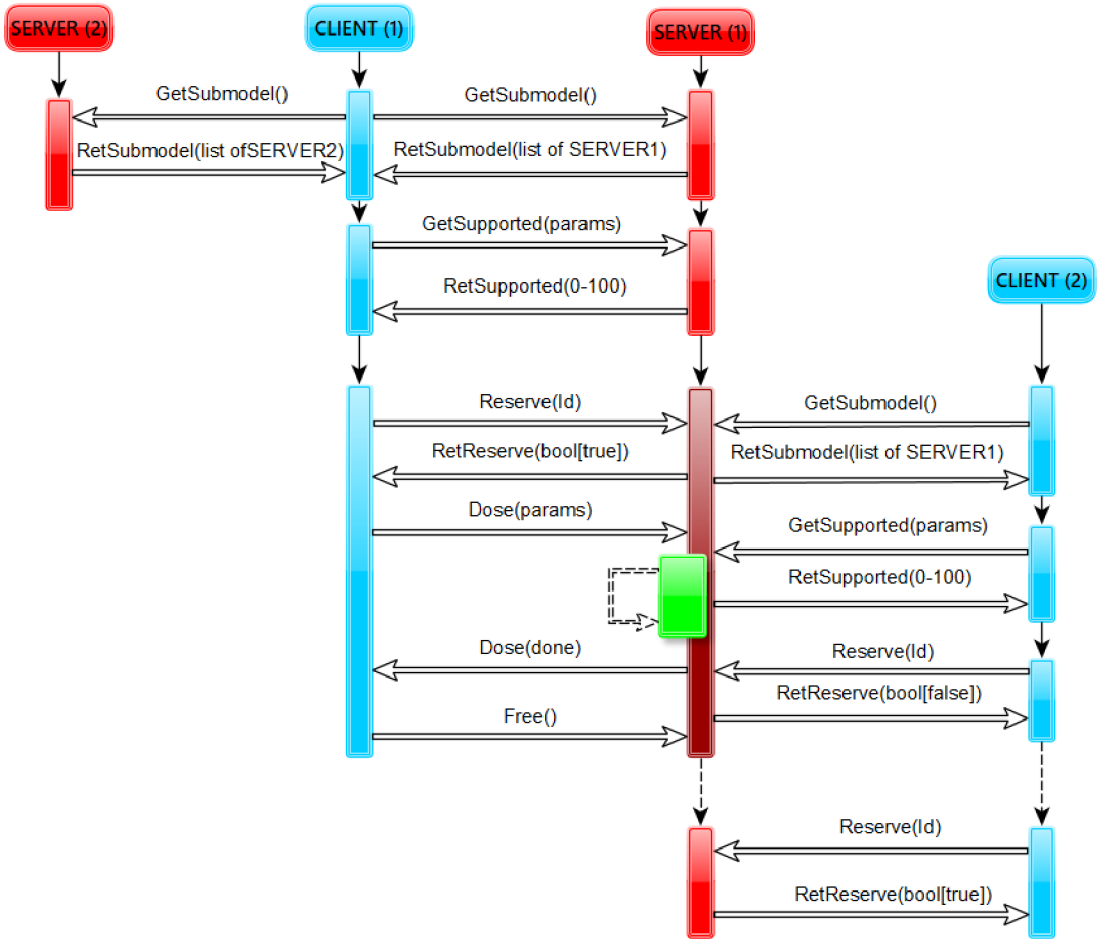
2.1.2 Komunikační vlastnosti AAS

Pro komunikaci AAS bylo zapotřebí přijít se samotným přístupem k toku dat ve výrobním procesu. Nakonec byl vybrán přístup produktového řízení výroby. Jinými slovy produkt zaštítěný pomocí AAS zadává příkazy ostatním AAS ve výrobním procesu, jenž mají za cíl vyrobit daný produkt. Produkt získá základní informace o požadavcích a postupech na jeho zhotovení při vytváření instance AAS konkrétního produktu.

Logickým postupem vychází, že komunikace tedy bude klient server, kde klient bude požadovatel služeb (výrobek) a server jejich poskytovatel (procesní jednotka). V ideálním případě by server měl mít možnost přechodu do stavu klient, jenž by umožňoval interakce procesních entit navzájem. Tato vlastnost serveru se ovšem v případě AAS pro operátora uvažovala až na další verzi zařízení.

komunikační diagram

Po ujasnění komunikačních vlastností byl proveden návrh samotného komunikačního toku dat viz. obrázek 2.1, jenž znázorňuje komunikační interakci dvou klientů a dvou serverů.



Obr. 2.1: Komunikační diagram AAS client server

Komunikace v ideální případě probíhá následovně:

- Klient zavolá službu GetSubmodel() - tímto mu server řekne o jaký typ procesní jednotky se jedná.
- Po ověření, že se jedná o správnou procesní jednotku, Klient zavolá službu GetSupported(params), kde parametry této služby jsou parametry hledané procesní služby, jenž se má poskytnout. Server vrací hodnotu 0 až 100 v závislosti na jeho schopnosti vykonávat danou procesní službu s požadovanými parametry.
- Pokud server vrátí hodnotu efektivního provedení procesní služby v mezích požadovaných Klientem, má klient možnost zavolání služby Reserved(id). Tato

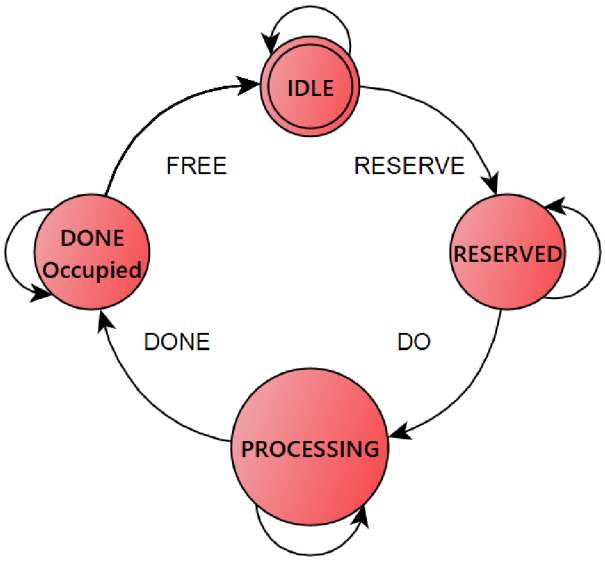
služba zarezervuje AAS serveru pro tohoto Klienta na základě jeho id. Služba Reserved(id) se provede pouze v případě, že AAS není již rezervováno jiným Klientem. Tato situace lze vidět na diagramu, kde Klient (2) se snaží rezervovat obsazený server (1).

- Následně si Klient pouze vyžádá provedení určité operace, která je podporovaná. Aktuální stav požadované služby Klient kontroluje pod dynamickou proměnnou.
- Po vykonání služby serverem Klient zavolá službu Free(id), jenž uvolní Server pro další operace.

2.1.3 Datové a aktivní vlastnosti AAS

U návrhu datových vlastností AAS se vycházelo z provedeného návrhu metamodelu AAS, jemuž je věnována kapitola viz. 2.3.

Aktivní vlastnosti AAS z vnějšího pohledu vychází z komunikačního návrhu a jejich vizualizaci lze vidět na stavovém automatu viz. obrázek 2.2.



Obr. 2.2: Hlavní stavový automat AAS

AAS obsahuje i interní aktivní části, jenž jsou zapotřebí například, pro řádnou operaci s daty, či pro rozhraní AAS s operátorem. Tyto části lze nalézt v podkapitole 2.3.3 a podkapitole 2.4.2.

2.2 Výběr vhodné HW a SW platformy pro AAS pro operátora

V zadání této práce nebyla specifikována požadovaná platforma, jenž by realizovala AAS pro operátora, z tohoto důvodu bylo zapotřebí provést zhodnocení všech dostupných HW platform, a z nich vyvodit nejvhodnější SW realizaci AAS pro operátora.

2.2.1 Výběr HW platformy

U výběru HW platformy lze uvažovat nepřeberné množství a variant, jenž by byly vhodnými kandidáty na výslednou implementaci. Výběr byl zúžen na tyto platformy: stolní počítač, notebook, vlastní embedded zařízení, chytrý telefon. V následujících částech bude proveden rozbor a zhodnocení již zmíněných platform.

Stolní počítač

Tato platforma obsahuje mnoho výhod pro implementaci AAS pro operátra. Jako hlavní výhody lze určitě považovat: velká dostupnost, přiměřená cenová nákladnost, možnost modularity, velikost výkonové i paměťové části a jednoduché a přehledné rozhraní pro uživatele.

Přes veškeré výhody, jenž tato platforma nabízí, nebyla vybrána jako vhodnou pro naši implementaci AAS pro operátora. Hlavním faktorem byla nevýhoda v podobě fixní pozice zařízení v prostoru.

Notebook

Notebook, oproti stolnímu počítači ztratil nevýhodu fixní pozice zařízení, a tudíž se jevil jako vhodný kandidát pro výslednou implementaci AAS.

I zde se ovšem objevily nevýhody, jenž byly důvodem neupřednostnění této platformy pro AAS. Jako hlavní nevýhodu lze považovat nutnost bezpečného prostoru pro odložení zařízení pro umožnění efektivní vykonávání úkonů operátora. Také bylo potřeba uvažovat robustnější, a tudíž i dražší variantu notebooku pro prodloužení životnosti zařízení z důvodu velké mechanické zatíženosti zapřičiněnou přenášením.

Embedded zařízení

U uvažování tohoto zařízení se dá hovořit například o vhodné variantě raspberry pi, integrovanou s pomocnými prvky v plastovém krytu. Tato platforma má pozitivní prvky modulovatelnosti i cenové dostupnosti. Modulovatelnost je ovšem omezená ve výkonu i datovém prostoru v porovnání například se stolním počítačem.

Jako hlavní nedostatky zde lze uvažovat náročnější HW kompletaci zařízení, či omezená možnost grafického rozhraní pro uživatele. A proto ani tato platforma nebyla zvolena pro implementaci AAS.

Chytrý telefon

Tato poslední uvažovaná platforma pro AAS se, i přes řadu nevýhod, zvolila jako nejvhodnější platforma pro realizaci AAS pro operátora.

Nevýhoda u této platformy nemožnosti HW modularity, zapříčiňuje omezený výběr komunikačních platforem, a tudíž i fixní výpočetní výkon i paměť zařízení. Také je zde nevýhoda nutnosti periodického dobíjení baterie zařízení.

Oproti tomu ovšem disponuje velkým množstvím výhod pro účel využití jako AAS pro operátora. Jako hlavní výhoda je bezesporu mobilita a skladnost zařízení, operátor má možnost jednoduchého uklizení zařízení (např. do kapsy na pracovním oděvu) za účelem uvolnění rukou pro vykonávání ostatních činností. Grafické rozhraní je zde přinejmenším postačující a v dnešním běžném životě často využívané. Také o této platformě lze hovořit jako o cenově dostupné.

Nevýhoda v podobě fixního výpočetního výkonu není pro implementaci AAS nikterak omezující, vzhledem k výkonům, jenž dosahují dnešní mobilní zařízení. Paměťové omezení lze kompenzovat využíváním síťových datových schránek či cloudů.

2.2.2 Výběr SW platformy

Po výběru HW platformy jako Chytrého telefonu, bylo nutné rozhodnutí výběru SW jenž bude AAS reprezentovat.

Jako první bylo zapotřebí vybrat operační systém (OS) chytrého telefonu, na kterém naše aplikace AAS bude fungovat. Uvažovat zde lze především OS. android, nebo OS. ios. Zde bylo rozhodnuto pro výběr OS. typu android pro jeho rozšířenost a jednodušší programovatelnost.

Výběr programovacího prostředí byl následně jednoduchý, neboť pro vývoj mobilních aplikací na OS. android již existuje programovací prostředí Android Studio. Vývoj byl z části ovšem proveden v prostředí CLion na OS. Linux a Windows, jednalo se především o programování Databáze a komunikačního rozhraní.

2.3 Návrh a realizace Datového modelu pro AAS

Při návrhu a realizaci Datového modelu pro AAS bylo vycházeno z kompromisů mezi již navrženými metamodeli jinými společnostmi (především společností ZVEI),

rozsahem nutné implementace a možností implementací databází na vícero platformách (jako např. PLC). Prvotně byl vytvořen návrh metamodelu AAS, jenž byl předlohou pro výslednou realizaci a implementaci databáze pro AAS pro operátora.

2.3.1 Návrh datového metamodelu

Návrh metamodelu viz. obrázek 2.3 prošel několika verzemi. Po každé verzi byla provedena konzultace s vedoucím práce na téma realizovatelnosti, praktičnosti a kompatibility. Výsledný metamodel je i, přes to značně abstraktní a idealizovaný oproti realizované databázi implementovaného AAS. Metamodel je také popisován v anglickém jazyce pro zajištění lepší technické čitelnosti a univerzálnosti jeho použití.

Jak již bylo zmíněno, při jeho návrhu metamodelu se vycházelo z teoretických znalostí získaných především z dokumentů od společnosti ZVEI. Jejich metamodel je ovšem velmi komplexní a využívá specifické datové typy s prvky, jenž by pro implemetnaci v AAS pro operátora byly velmi náročné a pro první verzi nadbytečné.

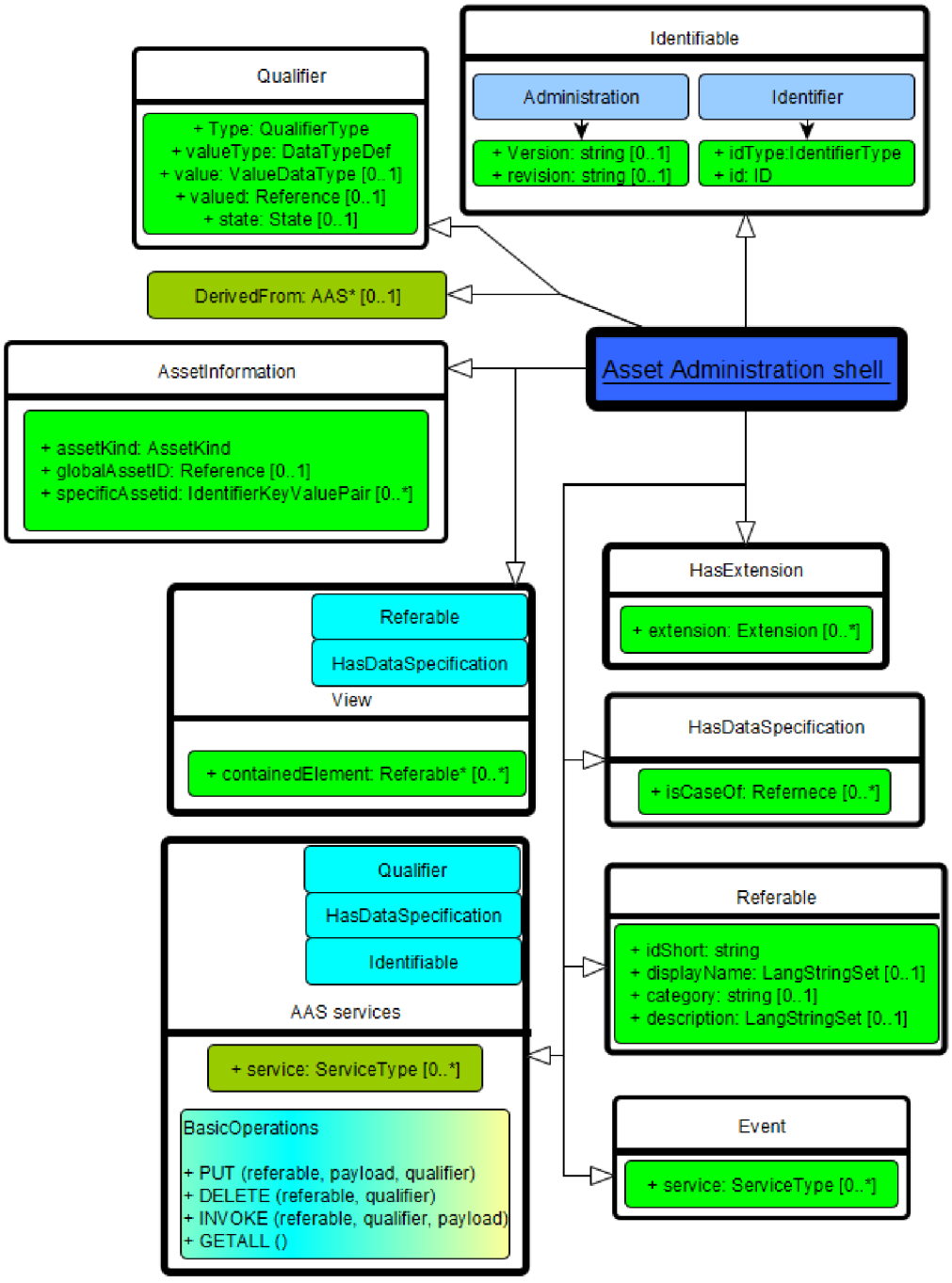
Nejdůležitějšími prvky v navrženém metamodelu jsou Identifable, AssetInformation, AAS services a HasDataSpecification. Následuje jejich stručný popis:

- Identifable - neboli identifikátor by měl obsahovat prvky, jenž zaručují bezpečnou a jednoznačnou identifikaci jednotlivých prvků spojených s AAS. Jedná se například o ID či administrativní data jako je například verze.
- AssetInformation - model představující základní informace o objektu jenž je zaštítěn pomocí AS.
- AAS services - tento prvek metamodelu byl navržen bez předchozí předlohy, neboť se jedná o návrh datového bloku pro specifické použití AAS. Nacházejí se zde ovšem prvky obecného použití, jenž nebyly následně implementovány.
- HasDataSpecification - zde lze nalézt rozšíření o jakýkoliv dodatečný informační obsah o jakémkoliv prvku. Výsledná implementace realizuje tento blok spojením vícero bloků dohromady pro zjednodušený přístup a realizaci.

2.3.2 Realizace datového modelu

Realizace datového modelu probíhala po částech při vytváření celkové databáze navázané na komunikaci. Jako programovací jazyk databáze byl zvolen C++. Jednotlivá data byla rozčleněna do často stejnojmenných tříd s realizovanými metodami pro operace s těmito daty. Díky této podobě implementace je výsledek velmi dobře modifikovatelný a lehce použitelný i na jiné instance AAS. Tato část kódu zaštiťuje veškerou pasivní část AAS.

Z důvodu rozsáhlosti této implementace bude následující pohled na realizovanou databázi rozčleněn na jednotlivé prvky, které budou popsány.



Obr. 2.3: Návrh metamodelu pro AAS

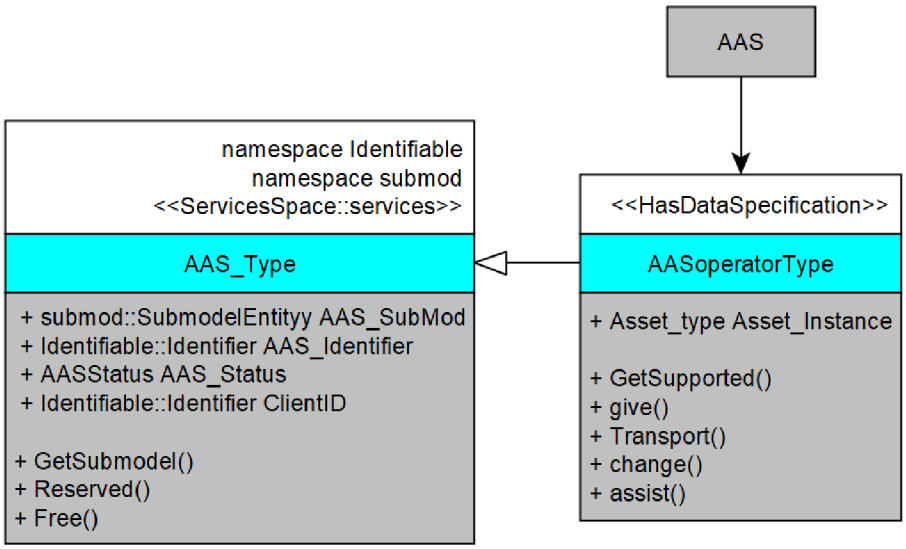
Obecně lze o všech diagramech zobrazující implementaci databáze AAS říci, že jsou popsány v anglickém jazyce ze stejného důvodu jako byl u návrhu metamodelu. Také se ve většině případů vyskytují metody, jenž poskytují funkce uložení a načtení dat (neboli getry a setry). U několika bloků je možno vidět v horní části řádky s označením "namespace", tyto řádky označují využité jmenné prostory z jiných částí kódu. Texty označené znaky: «...» realizují dědění z tříd o těchto názvech.

U diagramů jsou k nalezení případné odkazy na aktivní části databáze.

Základní Type a Instance AAS pro operátora

Jako první budou představeny základní třídy celého AAS viz. obrázek 2.4. AAS-Type je třída představující základ každé instance AAS, a z toho důvodu obsahuje většinu základních datových prvků a metod zaštiťující základní služby.

Třída AASoperatorType je základ pro všechny instance finálních AAS pro operátory vycházející ze základu AAS-Type. Obsahuje informace o zaštítěném objektu a specifické metody plnící funkci specifických služeb operátora.



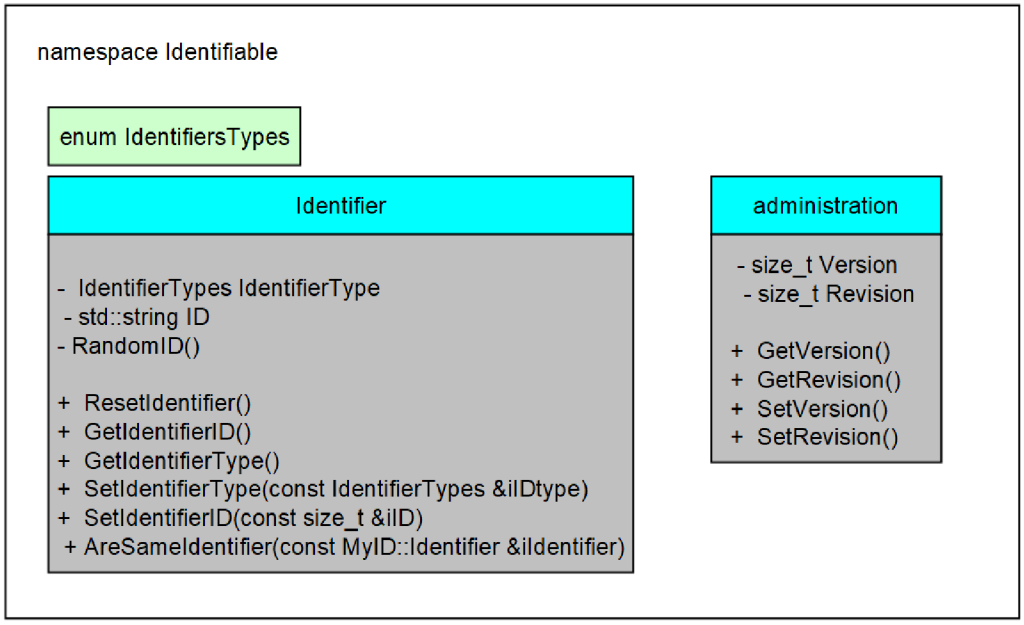
Obr. 2.4: Základní datový diagram Type a Instance AAS pro operátora

- třída AASoperatorType
  - Je třída, jenž dědí ze tříd HasDataSpecification a AAS-Type.
  - Datové proměnné: obsahuje datovou proměnnou typu Asset-type více popsanou v podkapitole 2.3.2.
  - Metody: obsahuje veřejné metody pro jednotlivé služby jenž podporuje operátor. Také se zde nachází metoda GetSupported() jenž je více popsaná v podkapitole 2.3.3.
- třída AAS-Type
  - Je třída, jenž dědí ze třídy ServicesSpace::services popsanou v podkapitole 2.3.2 a využívající jmené prostory Identifable (viz. podkapitola 2.3.2) a submod (viz. podkapitola 2.3.2).
  - Datové proměnné:
    - submod::SubmodelEntityy AAS-SubMod - Reprezentuje submodel AAS, který je zaštítěn AS více v podkapitole 2.3.2.

    - Identifiable::Identifier AAS-Identifier - je identifikátor AAS více v podkapitole 2.3.2.
    - AASStatus AAS-Status - je status aktuálního stavu AAS více v podkapitole 2.3.2.
    - Identifiable::Identifier ClientID - proměnná pro zapamatování si ID od klienta, jenž si vyžádal rezervaci AAS.
- Metody: Vyskytují se zde i klíčové metody zaručující správné fungování aktivní části AAS.
    - GetSubmodel() - metoda vracející submodel AAS.
    - Reserved() - metoda, která po zavolání plní operace nutné pro rezervování AAS více v podkapitole 2.3.3.
    - Free() - metoda sloužící k uvolnění AAS od rezervace více v podkapitole 2.3.3.

Datový blok Identifable

Datový blok Identifable je diverzifikován ve stejnojmenném jmeném prostoru Identifiable viz. obrázek 2.5. V tomto prostoru se nacházejí dvě datové třídy Identifier a administration. Také zde lze najít enumerační proměnnou IdentifiersType obsahující prvky: CUSTOM, IRI, IRDI a NULLPTR. Tato enumerační proměnná reprezentuje typy podporovaných ID.



Obr. 2.5: Datový diagram zobrazující jmenný prostor s třídami Identifier a Administration

- třída Identifier
  - Je třída reprezentující ID nejen AAS.
  - Datové proměnné:
    - IdentifierTypes IdentifierType - představuje typ reprezentace ID
    - std::string ID - je samotný identifikátor prvků AAS i AAS samotného.
  - Metody: krom metod nastavování a získávání dat se zde vyskytují i další tři metody.
    - RandomID() - metoda generující náhodné číselné ID, využívá se především u neinicializované konstrukce třídy Identifier.
    - ResetIdentifier() - metoda sloužící k vynulování ID a IdentifierType, používá se především u metody Free.
    - AreSameIdentifier() - metoda používaná k porovnání dvou tříd Identifier na základě jejich ID i IDentifierType.
- třída administration
  - Třída menšího významu s doplňujícími identifikačními prvky.
  - Datové proměnné: dvě jednoduché proměnné typu size-t
  - Metody: vyskytují se zde pouze metody pro nastavení a získání dat.

Datový blok AAS Services

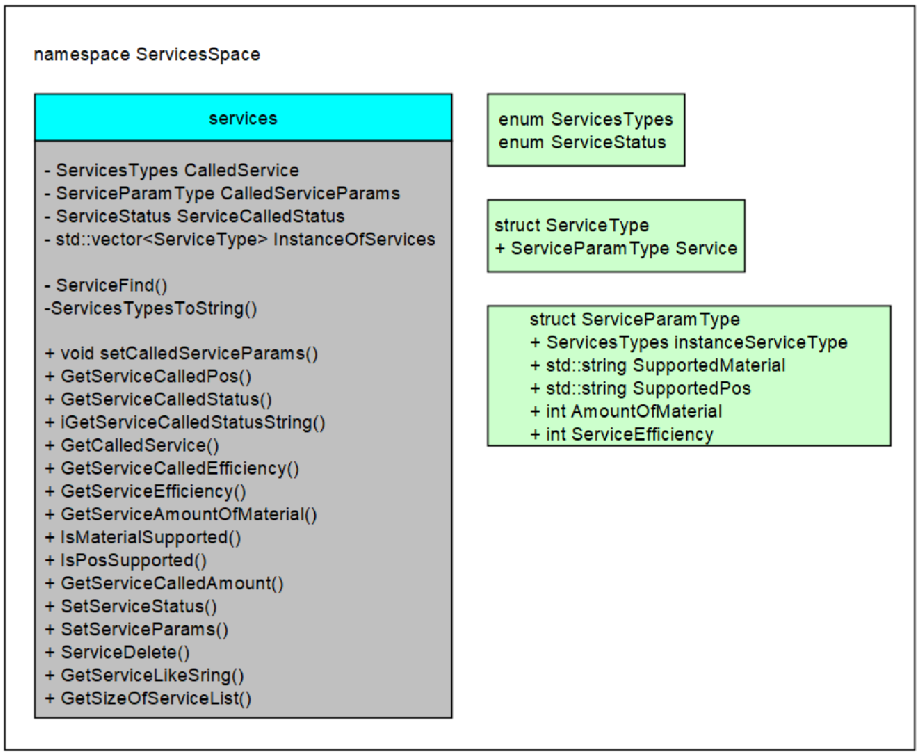
Tento datový blok se skládá ze dvou strukturu, dvou enumeračních proměnných a jedné třídy services vše zaštítěné jmenným prostorem ServicesSpace viz. obrázek 2.6.

Enumerační proměnná ServiceType reprezentuje typ služeb, jenž lze vykonávat pomocí AAS (GIVE, TRANSPORT, CHANGE, ASSIST, NONESERVICE) a druhá enumerační proměnná ServiceStatus obsahuje typy stavů, které může požadovaná služba nabývat (INPROGRESS, DONE, FAILED, NONE).

Struktura ServiceType obsahuje proměnnou Service typu struktura ServiceParamType. Struktura ServiceParamType obsahuje veškeré parametry služeb AAS.

Třída services

- Třída pracující s veškerými daty a interními operacemi, které se týkají služeb AAS.
- Datové proměnné:
  - ServicesTypes CalledService - jedná se o proměnnou do níž se ukládá typ služby, jenž je vyžadována.
  - ServiceParamType CalledServiceParams - proměnná typu ServiceParamType pro uložení parametrů volané služby.
  - ServiceStatus ServiceCalledStatus - proměnná sloužící pro ukládání stavu volané služby.

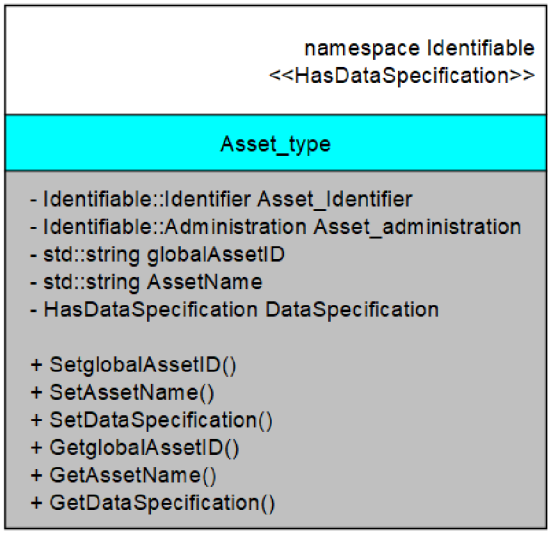


Obr. 2.6: Diagram datového bloku AAS Service

  - std::vector<ServiceType> InstanceOfServices - klíčová proměnná, jenž obsahuje veškeré uložené služby, které AAS podporuje.
- Metody: V této třídě se nachází nejvíce metod z celé databáze, je to zapříčiněno nejkomplexnějšími a nejrozsáhlejšími datovými prvky v AAS. Mezi velkým množstvím získávacích a nastavovacích metod se zde nachází i několik dalších metod, jenž budou popsány.
  - ServiceFind() - metoda sloužící pro nalezení pozice uložené služby ve vektoru InstanceOfServices.
  - ServiceDelete() - metoda používaná na nulování dat z vektoru InstanceOfServices.
  - GetServiceLikeSring() - metoda, která sestavuje std::string řetězec z parametrů obsahující služba nalezená ve vektoru InstanceOfServices.
  - GetSizeOfServiceList() - metoda obsahující v návratové hodnotě velikost vektoru InstanceOfServices.

Datový blok Asset information

Tento blok je reprezentován třídou Asset-type využívající jmenný prostor Identifiable a dědí z třídy HasDataSpecification viz. obrázek 2.7.

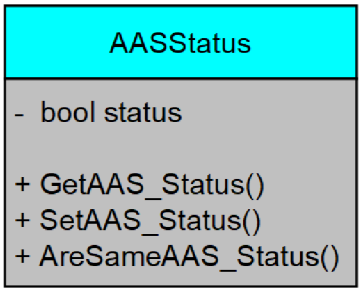


Obr. 2.7: Diagram datového bloku AAS Service

- Třída je zaměřena pouze na práci s pasivními daty objektu AS.
- Datové proměnné: vyskytují se zde instance tříd z jmenného prostoru Identifiable, a také tři proměnné specifické pro každého operátora. Jedná se o globalAssetID, AssetName a DataSpecification.
- Metody: metody zde obsažené jsou pouze určené k nastavování a získávání dat z privátní oblasti třídy.

Datový blok AAS status

Tento blok není nikterak vyobrazen na navrženém metamodelu AAS a byl přidán do implementace při vypracovávání databáze. Jedná se o prvek reprezentující stav AAS z pohledu rezervovanosti na základě bool proměnné viz. obrázek 2.8.



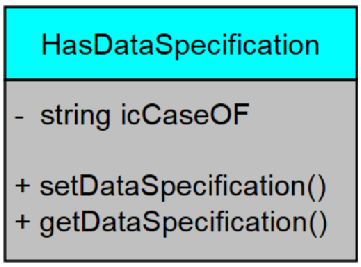
Obr. 2.8: Diagram datového bloku AAS Status

- Třída je velmi jednoduchá a obsahuje pouze jednu privátní proměnnou typu bool. Také obsahuje metody pro nastavování a získávání hodnoty proměnné a

metodu pro porovnávání dvou tříd AASStatus.

Datový blok AAS HasDataSpecification

Jedná se opět o méně klíčový datový blok, jenž je reprezentován stejnojmennou třídou HasDataSpecification viz. obrázek 2.9.



Obr. 2.9: Diagram datového bloku AAS HasDataSpecification

- Třída obsahuje jedinou proměnnou typu std::string jenž může reprezentovat referenci odkazující na další prvky či jinou formu rozšiřujících dat pro určitý prvek. Metody jsou čistě pro nahrávání a získávání těchto referenčních dat.

Datový blok AAS submodel

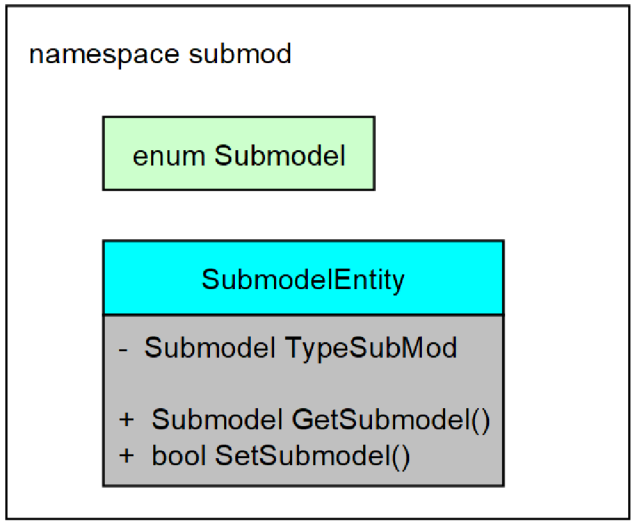
Tento datový blok obsahuje jmenný prostor submod, jenž zaštiťuje enumerační proměnnou a třídu SubmodelEntity. Enumerační proměnná obsahuje všechny typy submodelů, které se mohou vyskytovat v systému vněmž má být AAS implementováno (STORAGE, MANIPULATOR, MANUFACTURINGUNIT, OPERATOR, NULLPTR) viz. obrázek 2.10.

- Třída obsahuje jedinou proměnnou typu Submodel reprezentující typ submodelu, jenž je přiřazen k instanci AAS-Type. Metody plní funkci získávání a nahrávání dat.

2.3.3 Realizace aktivní části databáze

Realizovaná databáze AAS, krom pasivní části, která byla popsána, obsahuje také aktivní prvky v podobě metod, které pracují s privátními, nebo veřejnými daty.

Protože většina metod jsou realizací konstruktorů, destruktorů a nastavování a získávání hodnot, zde budou popsány a graficky zobrazeny diagramy klíčových metod pro funkčnost AAS.



Obr. 2.10: Diagram datového bloku Submodel

Metoda AAS-Type::Reserved

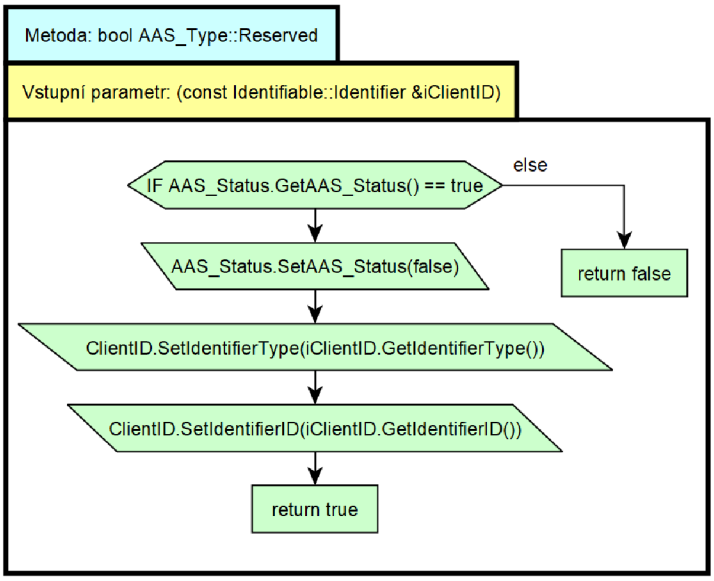
Metoda Reserved je klíčovou metodou ze základního návrhu stavového automatu AAS viz 2.2. Na obrázku 2.11 lze vidět diagram vykonávání této metody při jejím volání.

Vykonávání metody probíhá následovně: pokud není AAS zarezervováno, kontrola AAS statusu se uloží do proměnné ClientID typ clientova ID a jeho ID samotné. Metoda vrací bool hodnotu true nebo false na základě provedení podmínky, tato návratová hodnota je využívána pro detekci platného či neplatného pokusu o rezervaci AAS.

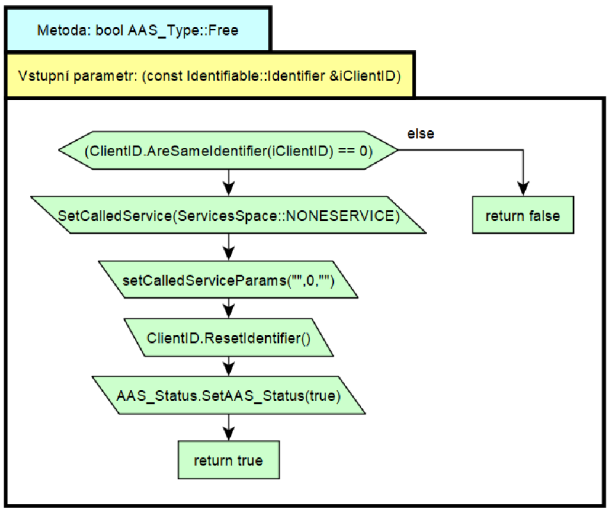
Metoda AAS-Type::Free

Tato metoda je také jednou z klíčových a její základní funkčnost lze najít ve stavovém automatu AAS viz 2.2. Diagram vykonávání této metody lze vidět na obrázku viz. 2.12.

Vykonávání metody probíhá následovně: jako první se provede kontrola identifikátoru žadatele o službu Free s Identifikátorem uloženým v ClientID, pokud tyto identifikátory nejsou souhlasné, pak se další část kódu již neprovádí. V případě, že identifikátory souhlasí, pak se provede nulování volané služby, resetování proměnné ClientID a nastavení statusu AAS na uvolněno (AAS-Status = true). Návratová hodnota metody je opět zpětnou kotrolou o provedení podmínky v kódu.



Obr. 2.11: Diagram metody AAS-Type::Reserved



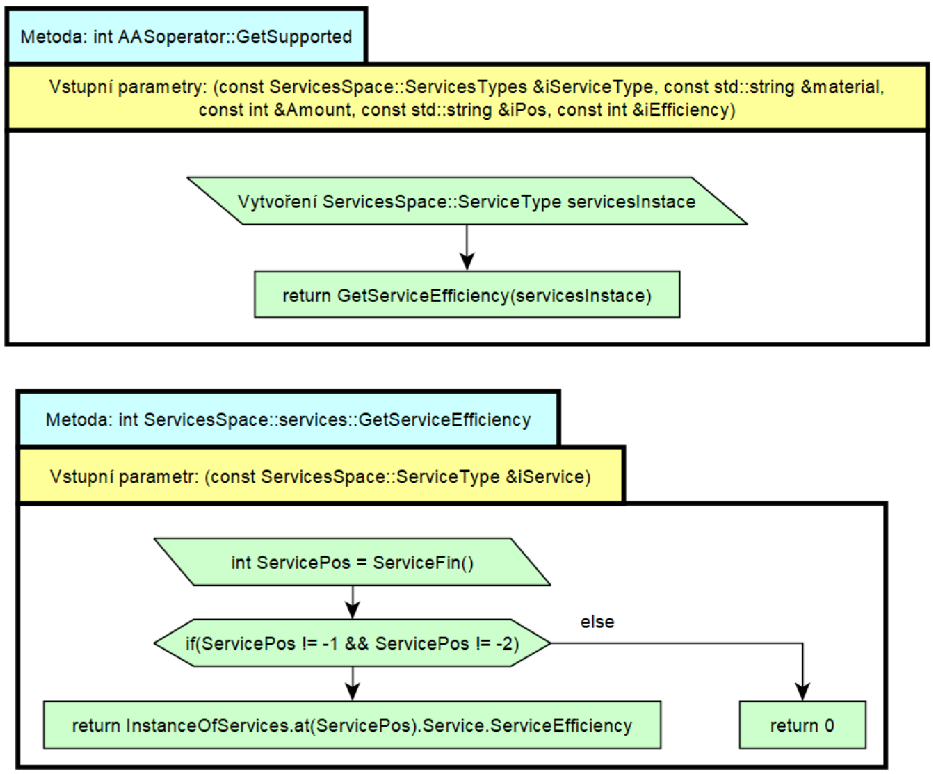
Obr. 2.12: Diagram metody AAS-Type::Free

Metoda AASoperator::GetSupported

I zde tato metoda vychází ze základního stavového autoamtu viz. 2.2. Metoda GetSupported využívá metodu z třídy services, konkrétně metodu GetServiceEfficiency diagramy těchto metod lze vidět na obrázku viz. 2.13.

Vykonávání metody probíhá následovně: Zavolání metody GetSupported se vytvoří proměnná typu ServiceType inicializovanou ze vstupních parametrů. Tato proměnná se používá jako vstupní parametr metody GetServiceEfficiency. Tato metoda ze třídy services, provede vyhledání služby ve vectoru uložených služeb. Pokud ta-

ková to služba je nalezena, metoda vrátí hodnotu efektivity vykonávání této služby, pokud není nalezena, pak vrátí základní hodnotu 0.



Obr. 2.13: Diagram metod GetSupported a GetServiceEfficiency

2.4 Výběr a realizace komunikace pro AAS

Návrh realizace, již proběhl v předchozích kapitolách, zde bude již proveden konkrétní výběr komunikační architektury a komunikačního protokolu. Také zde bude uvedena výsledná realizace těchto principů v AAS pro operátora.

2.4.1 Výběr komunikační architektury

Při výběru komunikační architektury byla možnost zvolit komunikační architekturu MQTT, nebo OPC UA. Vzhledem k decentralizované IoS povaze komunikace napříč jednotlivými AAS, byla snaho o výběr nejvhodnější architektury právě pro tuto implementaci.

MQTT architektura je založena na centrálních jednotkách řídící komunikaci. Tato vlastnost by se dala eliminovat redundantním přidáním těchto prvků. Nastávají zde ovšem komplikace v případěch, kdy v MQTT síti je velké množství zařízení, dochází k zahlcení sítě a k jejímu následnému kolapsu.

OPC UA architektura, oproti tomu, nemá centrální prvek, a ani nedochází k jejímu kolapsu při velkém množství zařízeních. Při velkém zatížení sítě dochází k nechtěným komunikačním latencím zpráv, jenž mohou dělat komplikace v časových intervalech komunikačních protokolů. Tyto latence jde z velké části eliminovat přidáním pomocných prvků, jako jsou například local discovery server (LDS) či multicast extension (ME).

Cílová implementace byla tedy zvolena jako OPC UA architektura, zatím bez LDS a ME prvků, neboť cílová síť neobsahuje kritické množství prvků. V případě nutnosti je možnost tyto prvky přidat do nových verzí produktu.

2.4.2 Realizace komunikace

Při implementaci architetkury OPC UA se zvolil komunikační protokol TCP. Jedná se o nejrychlejší duplexní komunikační protokol, který OPC UA podporuje.

Softwarová implementace byla provedena využitím veřejné knihovny napsané v programovacím jazyce C, jedná se o knihovnu Open62541, jenž má za snahu splňovat normu IEC 62541, která definuje OPC UA architekturu.

Programová implementace OPC UA

Z programovacího hlediska bylo vytvořeno rozhraní, jenž propojovalo C++ databázi s příkazy z Open62541.

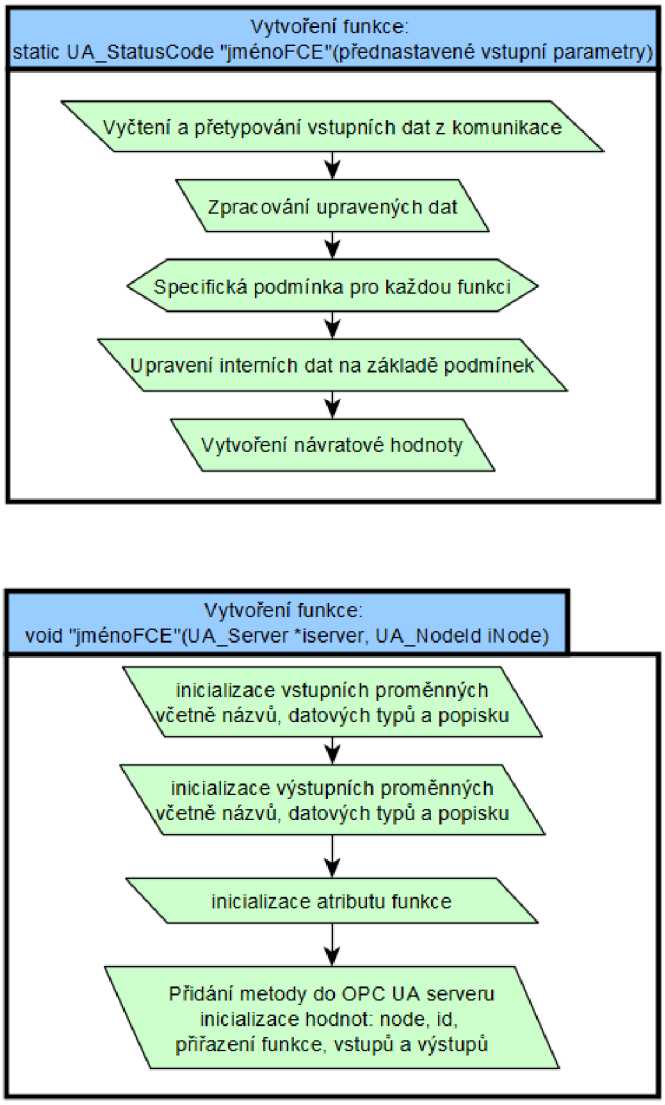
Jako první se musí inicializovat samotný server. Serveru se přiřadí komunikační socket a adresa. Adresa AAS v OPC UA se vždy inicializuje do následující podoby: opc.tcp://"ip adresa zařízení". Po spuštění komunikace se může zařízení podporující OPC UA a nacházející ve stejné síti připojit k AAS vyhledáním jeho adresy.

Vytvoření funkce/služby zpracovatelnou pro komunikaci OPC UA bylo zapotřebí každý prvek inicializovat. Tato inicializace probíhala ve dvou fázích, tyto fáze lze vidět na obrázku viz. 2.14.

Jako první fáze se vytvoří funkce typu void se vstupními parametry typu UA-Server a UA-Nodeld. Proměnná typu UA-Server slouží pro přiřazení funkce k danému serveru a UA-Nodel přiřazuje funkci pod příslušný uzel. Následně se provádí inicializace vstupních a výstupních proměnných a na závěr se provádí kompletace s referencí na funkci vykonávající samotné operace viz. druhá fáze.

Druhá fáze je vytvoření statické funkce návratového typu UA-StatusCode, tato funkce slouží pro vykonání jednotlivých operací při zavolání služby/funkce.

Největší komplikací u realizace druhé fáze inicializace služby/funkce, bylo přetypovávání univerzální vstupní proměnné typu UA-Variant na námi požadované

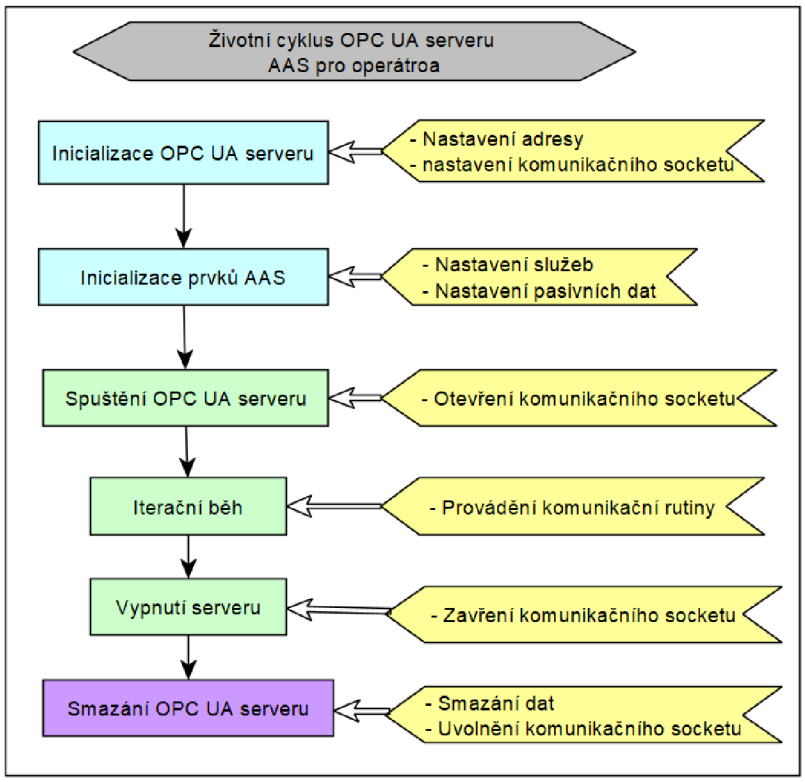


Obr. 2.14: Digram představující deklaraci a inicializaci služby/funkce pomocí Open62541

datové typy, neboť datové typy, s nimiž pracuje knihovna Open62541, jsou specificky vytvořené pro komunikační zpracování, zatímco databáze AAS je vytvořena pro obecné použití ve standardních datových typech.

Životní cyklus OPC UA v AAS pro operátora

Zde je představen zjednodušený pohled na operace, jenž se s OPC UA vykonávají pro zajištění komunikace viz. obrázek 2.15. Jednotlivé operace OPC UA serveru jsou rozděleny do metod v C++ knihovně, a jejich zavoláním se provádí daná operace.



Obr. 2.15: Digram životního cyklu OPC UA v AAS pro operátora

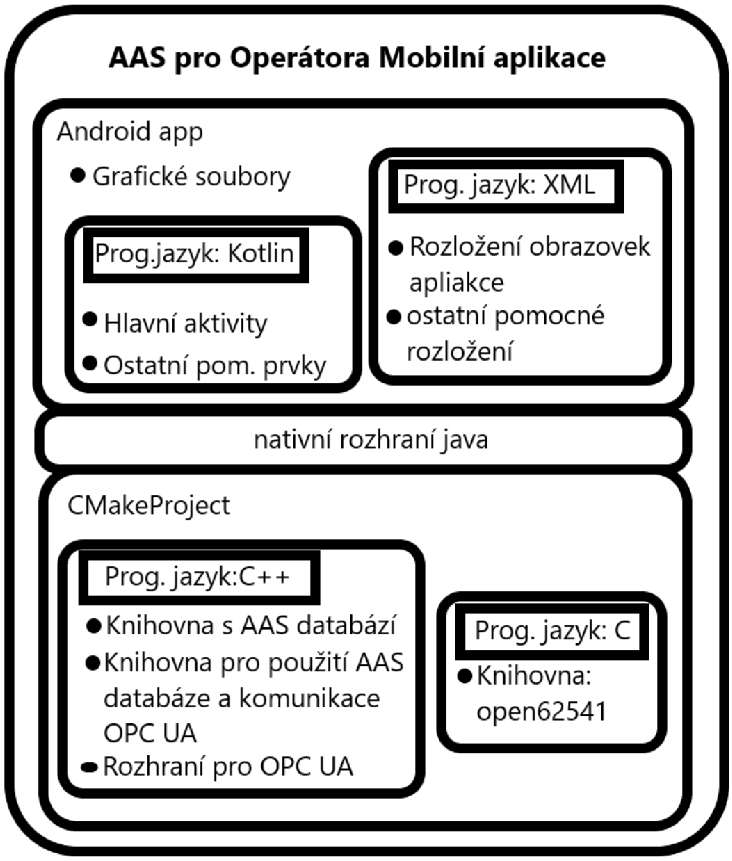
2.5 Realizace aplikace AAS pro Operátora na android zařízení

Zde bude představena nejkomplexnější část z celé práce, jedná se o kompletaci všech jednotlivých částí AAS do jednoho celku, a to mobilní aplikace pro OS android.

Vývoj aplikace byl prováděn v programovacím prostředí android studio s využitím programovacího jazyka kotlin. Vizualizace jednotlivých částí kódu lze vidět na obrázku viz. 2.16.

Tato android aplikace vyžaduje sadu pro vývoj softwaru (software development kit [SDK]) o minimální úrovni 21, což odpovídá android verzi 5 a výše. Na ostatních zařízeních není podporována. Cílené SDK je na nejnovější úroveň 32.

Realizaci jednotlivých částí projektu bude více rozebrána v následujících podkapitolách.



Obr. 2.16: Projektová vizualizace mobilní aplikace v android studiu

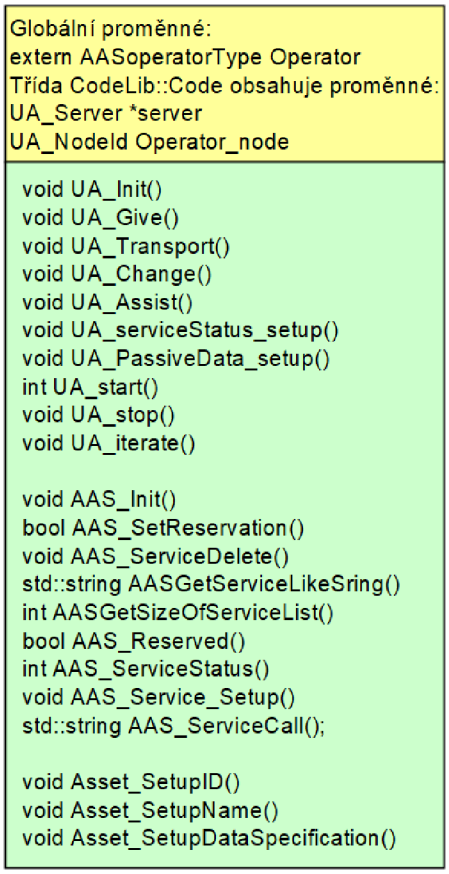
2.5.1 Realizace C++ knihovny

Tato část byla vyvíjena v prostředí CLion na OS linux. Základ tohoto projektu je CMake list (CMkakelist.txt vytváří automaticky specifický Makefile pro námi vytvořený kód).

Je zde obsažena komunikace a databáze AAS. Oba tyto prvky byly již představeny v předchozích kapitolách. Výsledný kód zde je zkompletován do použitelné knihovny a veškeré klíčové funkce se nacházejí v Code.cpp souboru. Seznam proměnných a funkcí, jenž tato knihovna poskytuje, je k vidění na obrázku 2.17. Knihovna je složena z jedné globální proměnné, představující instanci AAS operátora a třídy Code, obsahující veškeré metody pro práci s OPC UA serverem i databází AAS.

2.5.2 Vytváření rozhraní pro C++ a Kotlin

Pro možnost využití C++ knihovny v části kotlin kódu bylo zapotřebí vytvořit nativní knihovnu z jasně definovanými funkcemi, které tvoří nativní rozhraní java (Java Native Interface [JNI]). Podoba těcho funkcí musí být následovná: Datový typ



Obr. 2.17: Seznam proměnných a funkcí knihovny AAS

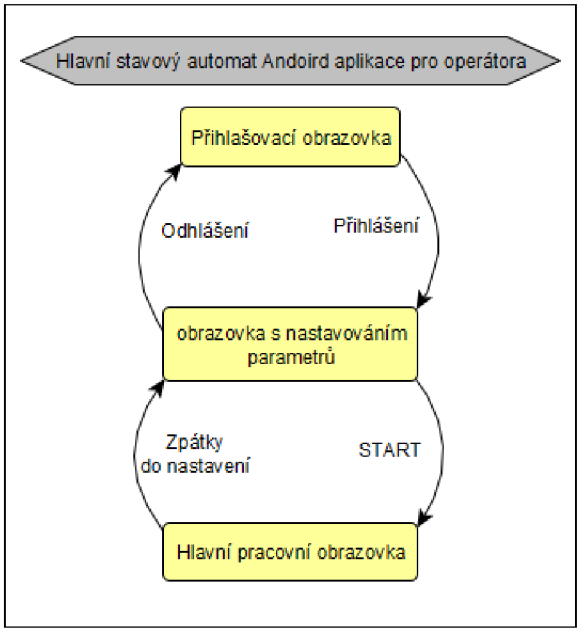
funkce je definován jako: extern "C"JNIEXPORT void JNICALL, a název funkce musí odpovídat cestě k aktivitě, v níž bude samotná funkce volaná například: Java-com-example-aascpp1-ServicesSettingUp-UAChange.

Také se zde muselo v několika místech řešit konflikt mezi standardními datovými typy C++ a objektovými typy Kotlinu. Tyto předy byly ovšem lehce vyřešeny.

2.5.3 Realizace Kotlin kódu

V samotném Kotlin kódu byl realizován kód vykonávající operace a řešící interakce se zadanými podněty z grafické části aplikace od uživatele/operátora. Kód se dělí do několika aktivit, kde každá aktivita odpovídá jedné grafické obrazovce, a také dvou pomocných Kotlin souborů sloužící pro interakce s výčtovým listem, jenž se v aplikaci nachází.

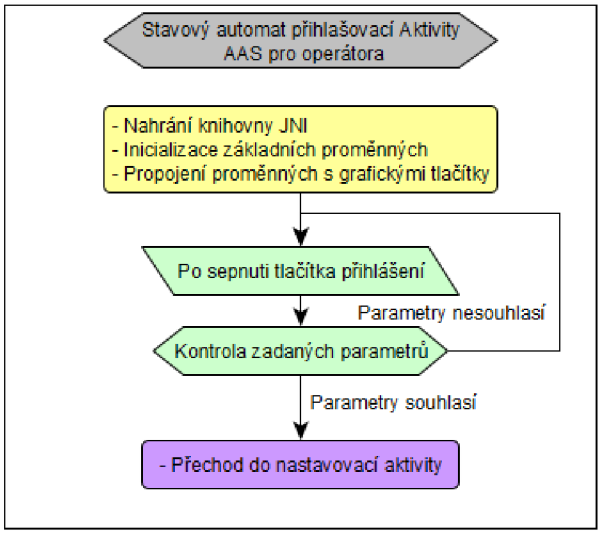
Mezi jednotlivými aktivitami/obrazovkami jsou vazby, jenž znázorňuje stavový automat viz. obrázek B.2.



Obr. 2.18: Hlavní stavový automat Mobilní aplikace

Přihlašovací aktivita (v projektu: MainActivity)

Tato aktivita je přímo vztažená k přihlašovací obrazovce viz. obrázek 2.20. Je zde pouze řešena zobrazování ikon a kontrola podmínek a přihlašovací parametrů viz. obrázek 2.19. Také jsou zde znázorněny základní informace o aplikaci.



Obr. 2.19: Hlavní stavový automat přihlašovací aktivity

Aktivita s nastavováním parametrů (v projektu: ServicesSettingUp)

Jedná se o aktivitu, která se zabývá nastavováním a spravováním dat od uživatele. Zpracovávají se zde zadané parametry služeb, jenž jsou v požadavcích přidání do databáze AAS, a také zde lze nastavit několik statických dat o uživateli (objektu AS). Statická data se zadávají pomocí vyskakovacího okna.

Diagram této aktivity lze vidět na obrázku v příloze viz. A.4. S touto aktivitou souvisí soubory ClassAdapter a Service, jedná se o pomocnou třídu, jenž má za úkol reprezentovat a ukládat data pro uživatelský posuvný list poskytovaných služeb. Tento list se s každým spuštěním aktivity načte z paměti AAS. Pro zaručení neztracení přehledu o nastavených službách při přechodu mezi aktivitami. Tento list také umožňuje uživateli označit jednotlivé služby, a ty vymazat, nejen z grafické části, ale i z paměti AAS.

Hlavní pracovní aktivita (v projektu: RunScreen)

Pracovní aktivita je nejdůležitější aktivitou v mobilní aplikaci. Uživatel, jenž se nachází v této aktivitě/obrazovce, je v pracovním režimu a jeho AAS je připojeno do sítě jako OPC UA server.

Nachází se zde několik vláken kódu.

- první a hlavní vlákno se spustí okamžitě při běhu a obstarává hlídání změn tlačítek, inicializace proměnných a spouští ostatní vlákna.
- Druhé vlákno slouží pro iteraci OPC UA komunikace a běží neustále, pokud je splněna podmínka, jenž je stále true.
- Třetí vlákno se spouští a zastavuje stejně jako druhé vlákno, řeší ovšem změny proměnných a stavů AAS zapřičiněných komunikací či uživatelem.

Diagram této aktivity lze vidět na obrázku v příloze viz. A.1, a třetí komplexnější vlákno je znázorněno diagramem v příloze na obrázku A.2.

2.5.4 Realizace grafického rozhraní

Grafické rozhraní bylo realizováno v podobě xml souborů, jenž android studio zpracovává na obrazovku mobilního zařízení. V těchto souborech bylo nutné nadefinovat typy grafických prvků, jejich vlastnosti, identifikátory a poziční vazby.

Mimo hlavních obrazovek se v projektu vyskytují ještě dva další xml soubory. Xml soubor item-service.xml realizuje zobrazovaný prvek v grafickém listu služeb v nastavovací obrazovce a soubor activity-services-setting-up.xml je grafickou reprezentaci vyskakovacího okna pro nastavování statických parametrů o uživateli (objektu AS).

Výsledné grafické obrazovky lze vidět v kapitole 2.6.1.

2.6 Testování aplikace AAS pro operátroa

Testování výsledné aplikace AAS pro operátora probíhalo ne dvou zařízení s OS. Android. Konkrétně se jednalo o Xiaomi Redmi Note 10 5G a Xiaomi Redmi Note 7. Jako client pro simulaci komunikace sloužil notebook s nainstalovaným programem UaExpert.

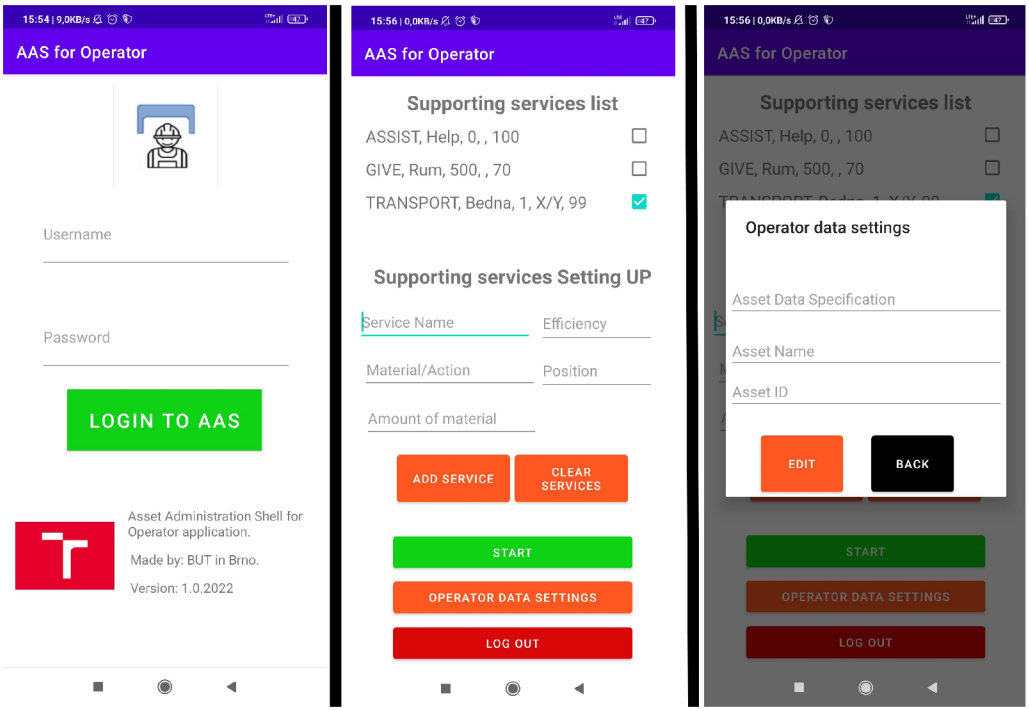
Při testování byla ověřována funkčnost grafického rozhraní, operací s databází AAS a Komunikace.

2.6.1 Testování grafického rozhraní

Testování grafického rozhraní se soustředilo především na dynamicky se měnící texty, správnou funkčnost tlačítek a akcí, jenž následují a zadávání a zpracování textů do textových polích uživatelem.

Grafické zobrazení aplikace lze vidět na obrázích viz. 2.20 a 2.21.

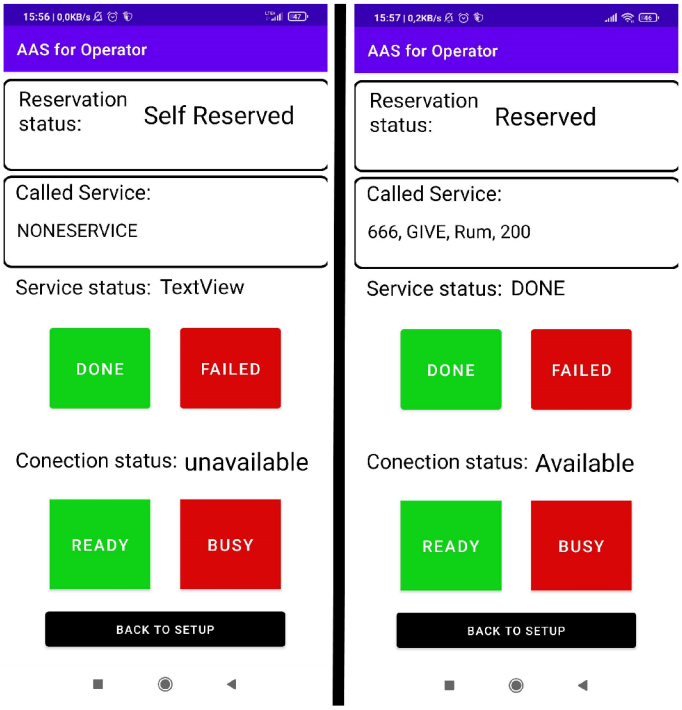
Na obrázku 2.20 v levé části lze vidět přihlašovací obrazovka. Uprostřed obrázku se nachází nastavovací obrazovka a úplně napravo je zobrazeno vyskakovací okno s nastavováním dat o operátorovi. V nastavovací obrazovce lze vidět v horní části list služeb, které již obsahuje vytvořené služby. Z těchto služeb je služba TRANSPORT označena a po aktivování tlačítka CLEAR SERVICES by došlo k jejímu odstranění.



Obr. 2.20: Hlavní stavový automat Mobilní aplikace

Na obrázku 2.21 je vyobrazena pracovní obrazovka spjata s pracovní aktivitou. Nalevo se jedná o zobrazení obrazovky ihned po přechodu do této aktivity. Napravo se již jedná o ukázku pracovní obrazovky po zarezervování a zavolání služby GIVE. Také si lze všimnout, že v obrazovce, jenž se nachází v rezervaci, je již operátorem vykonána aktivita a status služby je nastaven na DONE.

Grafické prvky v mobilní aplikaci AAS pracují dle vykonaných testů bez problému.



Obr. 2.21: Hlavní stavový automat Mobilní aplikace

2.6.2 Testování databáze AAS

Testování databáze probíhalo paralelně s testováním všech ostatních prvků. Při testování bylo ověřeno správné zacházení s daty AAS, volání jednotlivých služeb i práce s daty služeb.

Také se zde vytvořili konfliktní stavy, jenž měly za úkol otestovat chování aplikace při nedefinovaných stavech. Tyto simulace vždy proběhli dle očekávání a databáze AAS si s nimi dokázala poradit.

2.6.3 Testování komunikace

Jak již bylo zmíněno pro testování komunikace byl využit program UaExpert. S tímto programem bylo navázáno spojení a na modelových situacích se prováděla simulace

komunikace client server v souladu s komunikačním návrhem. Všechny tyto testy vyšly úspěšně. Na následujících obrázcích jsou vyobrazeny volání jednotlivých služeb a jejich výsledek se specifickým nastavením AAS.

2.6.4 Výsledky testování

Z komplexního testování mobilní aplikace AAS pro operátora, se přišlo na jeden problémový stav aplikace. Pokud uživatel přejde z pracovní obrazovky/aktivity do nastavovací obrazovky a zpátky, pak se již UaExpert nebyl schopen znovu připojit k OPC UA serveru AAS. A při opakovaném zkoušení o připojení aplikace vyústila v nouzové vypnutí (tzv "spadla"). Tento jev je způsoben nedodržením přesného postupu u životního cyklu OPC UA serveru. OPC UA Server AAS totiž neprovádí smazání serveru. Smazání serveru provede uvolnění komunikačního socketu, tuto operaci aplikace AAS pro operátora nevykonává z důvodu vyvolání automatického spadnutí aplikace ihned o pokusu zavolání této funkce. Vyvolání této fatální chyby po snaze smazat OPC UA Server se nepodařilo objasnit.

Při testech komunikačního životního cyklu v prostředí Linux se tento problém nevyskytoval a vše se vykonávalo dle očekávání. Z tohoto důvodu lze usuzovat, že se jedná o problém, jenž sem zanáší OS. android. Nepodařilo se ovšem žádným způsobem zjistit příčina, a proto by to mělo být ve snaze vyřešit v nadcházející verzi aplikace.

2.7 Návrh druhé verze AAS pro operátora

2.7.1 Problémy a nedostatky v aktuální verzi

Hlavní problém je s implementací komunikačního životního cyklu v mobilní aplikaci. Tento problém omezuje funkčnost aplikace, nezamezuje ji ovšem v jejím používání seznámeným operátorem.

Mezi nedostatky se dá považovat absence ukládání dat do trvalé paměti zařízení. Jedná se především o data nastavených služeb uživatelem. Také zde ovšem není možnost registrace v přihlašovací obrazovce. Tento nedostatek je spojen s ukládáním dat do dlouhodobé paměti v zařízení.

2.7.2 Možné budoucí rozšíření

Množství rozšíření AAS pro operátora je velmi mnoho, takřka každá část AAS pro operátora lze rozšířit a vylepšit. Z důvodu velké komplexnosti této problematiky nebyl na tyto rozšíření a vylepšení časový prostor.

V oblasti databáze AAS lze rozšířit jednotlivé prvky o další modely z návrhu metamodelu AAS. Také by bylo vhodné zvážit přidání fronty na požadavky rezervace AAS. Tato vlastnost by byla jednoduše realizovaná změnou přístupu ke komunikační hierarchii a vytvoření vektorové proměnné s požadavky na rezervaci.

U komunikační části OPC UA v AAS lze uvažovat nad přidáním možnosti změny statusu Server na client pro schopnost AAS požadovat služby od ostatních AAS.

Mobilní aplikace vyžaduje vyřešení problému a nedostatků, jenž byly popsány výše. Bylo by též vhodné implementovat elektronickou podobu návodu na obsluhu aplikace.

Závěr

V teoretickém rozboru byly splněny cíle popsání AAS a jeho částí. AAS bylo popsáno jako virtuální administrativní schránka pro decentralizované systémy I4.0, včetně historie jeho vývoje, životního cyklu a příklady přístupů k AAS v RAMI 4.0 modelu.

Také byla definována pasivní část AAS, do niž patří vícero druhů paměťových oblastí a využívání souborů pro jednotný přístup k manipulaci s AAS. Oproti tomu aktivní část AAS charakterizuje soubor vlastností pro interakci s pasivními daty a komunikační schopnosti AAS napříč různými druhy systémových komunikačních sítí.

Při návrhu AAS pro operátora byl proveden návrh vhodné komunikační architektury a hardwarové platformy pro realizaci AAS. Komunikační architektura byla zvolena OPC UA s TCP protokolem. Tato architektura byla následně implementována pomocí veřejné knihovny Open62541. Jako hardwarová platforma byl vybrán chytrý telefon s OS. Android s minimální verzí Android 5.

Pasivní i aktivní část AAS byla realizována jako univerzální knihovna v programovacím jazyce C/C++. Díky tomuto je možnost implementovat toto řešení i na jiných operačních systémech, jako jsou například Linux či Windows, u těchto platforem jsme ovšem omezeni doposud neexistujícím uživatelským rozhraním.

Uživatelské rozhraní bylo realizováno pomocí Android studia v programovacím jazyce Kotlin. Výsledná mobilní aplikace realizuje AAS pro operátora ve funkční verzi pro poučenou obsluhu. Toto poučení lze zprostředkovat seznámením se s napsaným návodem k aplikaci s ohledem na kompatibilitu s ostatními formami implementace školního standardu AAS.

Výsledné AAS bylo řádně testováno, nalezené problémy a nedostatky byly zhodnoceny a uvedeny. Také byly zváženy možnosti rozšíření aktuálního AAS pro operátora o další prvky a vlastnosti.

Literatura

[1] *Jak rozumět konceptu Průmysl 4.0. Svaz průmyslu a dopravy České republiky [online]. (ČR): Svaz průmyslu a dopravy České republiky, 2019, 19.8.2019 [cit. 2021-10-28].* Dostupné z URL: <https://www.spcr.cz/aktivity/z-hospodarske-politiky/12973-jak-rozumet-konceptu-prumysl-4-0>.

[2] *C. Wagner et al., "The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant,"2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2017, pp. 1-8, doi: 10.1109/ETFA.2017.8247583.*

[3] *RUDLA, Bc. Jakub. Aktuální problémy implementace principů Průmyslu 4.0 [online]. Plzeň, 2018 [cit. 2021-11-14].* Dostupné z URL: <https://dspace5.zcu.cz/handle/11025/31346.DIPLOMOVÁPRÁCE.ZÁPADOČESKÁUNIVERZITA>. DIPLOMOVÁ PRÁCE. ZÁPADOČESKÁ UNIVERZITA V PLZNI FAKULTA STROJNÍ. Vedoucí práce Doc. Ing. Jan HOREJC, Ph.D.

[4] *Big Data What it is and why it matters. SAS [online]. Stockholm, Sweden: SAS, - [cit. 2021-11-14].* Dostupné z URL: <https://www.sas.com/en_us/insights/big-data/what-is-big-data.html>.

[5] *ZEZULKA, František, Petr FIEDLER a Zdenek BRADÁČ. Prostředky průmyslové automatizace. FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, 2002.*

[6] *ZEZULKA, František, Ivo VESELÝ a Vlastimil BRAUN. Úvod do problematiky a základní modely Industry 4.0. SystemOnLine [online]. -: IT Systems, 2017, 2017 [cit. 2021-11-14].* Dostupné z URL: <https://www.systemonline.cz/rizeni-vyroby/uvod-do-problematiky-a-zakladni-modely-industry-4.0.htm>.

[7] *ARM, Jakub, Tomas BENESL, Petr MARCON, et al. Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0. Sensors. 2021, 2021(6), 1-20. ISSN 1424-8220.* Dostupné z: doi:10.3390/s21062004

[8] *Industrie 4.0 component and the concept of the Asset Administration Shell. Standartizaton council Industrie 4.0 [online]. Německo: -, 2017 [cit. 2021-11-20].* Dostupné z URL: <https://www.sci40.com/english/asset-administration-shell/>.

[9] *VERZANO, Nemrude. Interoperability – The Key Enabler of Industrie 4.0. SAP [online]. Německo: -, -, 20.11.2019 [cit. 2021-11-20].* Dostupné z URL: <https://blogs.sap.com/2019/11/20/interoperability-the-key-enabler-of-indust 0/>.

[10] *Barnstedt, E and Bedenbender, H and Billmann, M and Boss, B and Clauer, E and Fritsche, M and Garrels, K and Hankel, M and Hillermeier, O and Hoffmeister, M and others. Details of the Asset Administration Shell Part 1. Technical report, Federal Ministry for Economic Affairs, 2020.*

[11] *RÜDIGER, Fritz, Dr. Andreas Graf GATTERBURG„ Michael RIESTER a Marvin WIEGAND. The Asset Administration Shell in the OI4 solution framework. Open Industry 4.0 Alliance [online]. Německo: Open Industry 4.0 Alliance, 2019, 22.července.2021 [cit. 2021-11-28].* Dostupné z URL: <https://openindustry4.com/dam/jcr:6048c20f-4e09-4fa5-b07c-1411b2e03f0c/ The%20Asset%20Administration%20Shell%20in%20the%20OI4%20solution% 20framework%20v1.0%2020210722.pdf>.

[12] *Details of the Administration Shell - from idea to implementation. Plattform-i40.de [online]. Německo: BMWI, 2013, 1.7.2019 [cit. 2021-11-28].* Dostupné z URL: <https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/ Publikation/vws-in-detail-presentation.html>.

[13] *SEIF, Alejandro, Carlos TORO a Humza AKHTAR. Implementing Industry 4.0 Asset Administrative Shells in Mini Factories. Procedia Computer Science [online]. 2019, 2019, 2019(159), 495-504 [cit. 2021-11-28]. ISSN 1877-0509.* Dostupné z URL: <doi:https://doi.org/10.1016/j.procs.2019.09.204>.

[14] *Bedenbender, H and Bentkus, A and Epple, U and Hadlich, T and Heidel, R and Hillermeier, O and Hoffmeister, M and Huhle, H and Kiele-Dunsche, M and Koziolek, H and others. Industrie 4.0 plug-and-produce for adaptable factories: Example use case definition models and implementation. Federal Ministry for Economic Affairs and Energy (BMWi). Německo, 2017, x(x), 1-69.*

[15] *PROFANTER, Stefan, Kirill DOROFEEV, Alois ZOITL a Alois KNOLL. OPC UA for plug & produce: Automatic device discovery using LDS-ME. International Conference on Emerging Technologies and Factory Automation (ETFA). 2017, 22, 1-8.*

[16] *YANG, Wen, Haider, Naeem, ZOU, Jian-hong, Zhao a Qianchuan. Industrial Big Data Platform Based on Open Source Software. 2017. Dostupné z: doi:10.2991/cnct-16.2017.90*

[17] *BOSS, Birgit, Somayeh MALAKUTI, Shi-Wan LIN (YO-I), Thomas USLÄNDER, Erich CLAUER, Michael HOFFMEISTER a Ljiljana STOJANOVIC. Digital twin and asset administration shell concepts and application in the industrial internet and industrie 4.0. Plattform Industrie. 2020, (4), 1-33.*

[18] *Hu, Vincent and Ferraiolo, David and Kuhn, D. and Schnitzer, A. and Sandlin, Knox and Miller, R. and Scarfone, Karen. Guide to attribute based access control (ABAC) definition and considerations. National Institute of Standards and Technology Special Publication. 2014, (1), 162-800.*

[19] *Bader, Sebastian and Berres, Bernd and Boss, Birgit and Gatterburg, Andreas and Hoffmeister, Michael and Kogan, Yevgen and Köpke, Alexander and Lieske, Matthias and Miny, Torben and Neidig, Jörg and Orzelski, Andreas and Pollmeier, Stefan and Sauer, Manuel and Schel, Daniel and Schröder, Tizian and Thron, Mario and Usländer, Thomas and Vialkowitsch, Jens and Vollmar, Friedrich and Ziesche, Constantin.Details of the Asset Administration Shell. Part 2 -Interoperability at Runtime - Exchanging Information via Application Programming Interfaces (Version 1.0RC01). Technical report, Federal Ministry for Economic Affairs, 2020.*

[20] *PÁSEK, Jan. Digitální transformace průmyslu. FEKT VUT v Brně.*

Seznam symbolů a zkratek

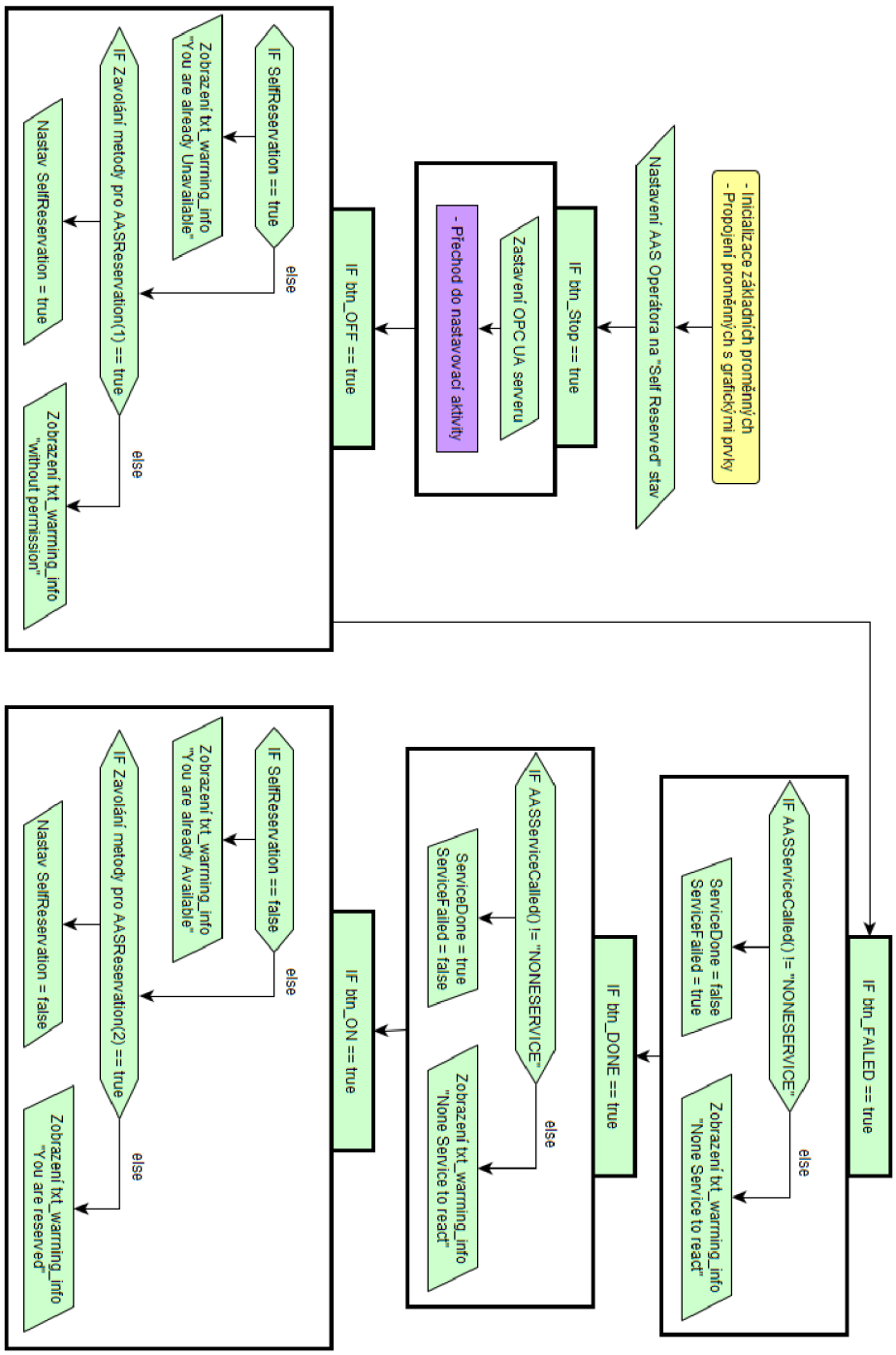
I4.0	Industry 4.0 – Průmysl 4.0
AAS	Asset Administration Shell – Objektová Administrativní Schránka
AS	Administration Shell – Administrativní Schránka
AI	Artificial Intelligence – umělá inteligence
IoT	Internet of Thinks – Internet věcí
IoS	Internet of Services – Internet služeb
IoP	Internet of People – Internet lidí
SOA	Service Oriented Architectures – servisově orientovaná architektura
MES	Manufacturing Execution System – Výrobní prováděcí systém
ERP	Enterprise Resource Planning – Plánování podnikových zdrojů
HW	Hardware
SW	Software
RAMI 4.0	Reference Architectural Model Industrie 4.0 – referenční architektonický model I4.0
IIC	Industrial IoT Reference Architecture – Referenční architektura průmyslového IoT
DT	Digital Twin – digitální dvojče
API	Application Programming Interface – Aplikační programovací rozhraní
TC	Technical Committee – Technický výbor
WG	working group – pracující skupina
IEC	International Electrotechnical Commission – Mezinárodní elektrotechnická komise
ISO	International Organization for Standardization – Mezinárodní organizace pro normalizaci
SCI 4.0	Standartizaton council Industrie 4.0 – Standardizační rada I4.0

SAP	Systems - Applications - Products in data processing – Systémy - Aplikace - Produkty ve zpracování dat
EU	European Union – Evropská unie
ID	Identifier – Identifikátor
OPC UA	Object Process Control Unified Architecture – Jednotná architektura řízení objektových procesů
IRDI	International Registration Data Identifier – Mezinárodní registrační datový identifikátor
IRI	Internationalized Resource Identifier – Internacionalizovaný identifikátor zdroje
URI	Uniform Resource Identifier – Jednotný identifikátor zdroje
URL	Uniform Resource Locator – Jednotný lokátor zdroje
RFC	Request for Comments – Žádost o komentáře
MQ	Message Queue – Fronta zpráv
MQTT	MQ Telemetry Transport – Telemetrická doprava MQ
TCP/IP	Transmission Control Protocol/Internet Protocol – přenosový kontrolní protokol/Internetový protokol
LDS	local discovery server – místní vyhledávací server
ME	(multicast extention – rozšíření multicast
SR	service requester – žadatel o službu
SP	service provider – Poskytovatel služeb
ABAC	Attribute-based access control – Řízení přístupu založené na atributech
XML	Extensible Markup Language – Rozšiřitelný značkovací jazyk
json	JavaScript Object Notation – Zápis objektů JavaScript
DRM	Digital rights management – Správa digitálních práv

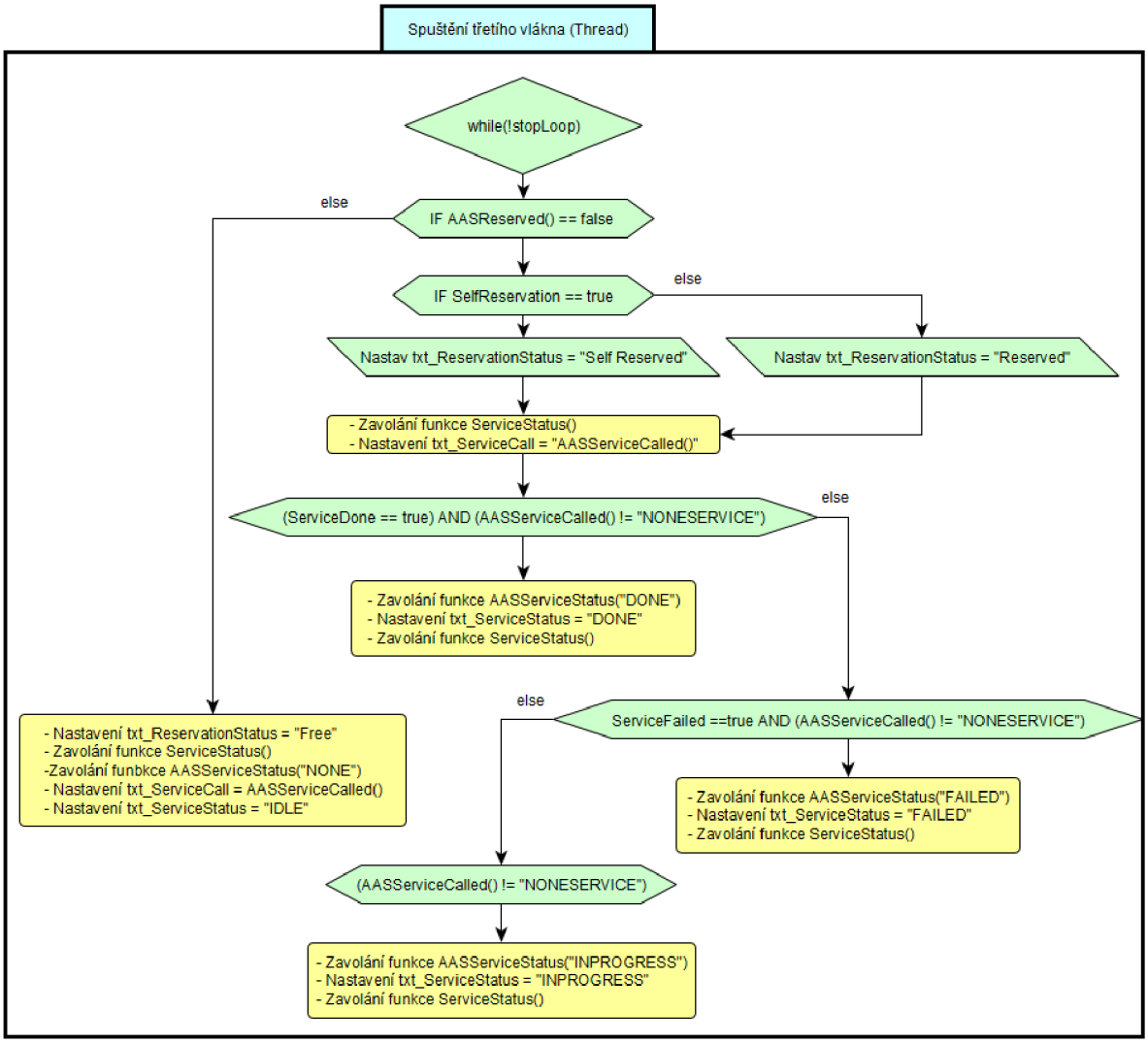
Seznam příloh

A Diagramy kotlin kódu

A.1 Hlavní Pracovní aktivita/obrazovka

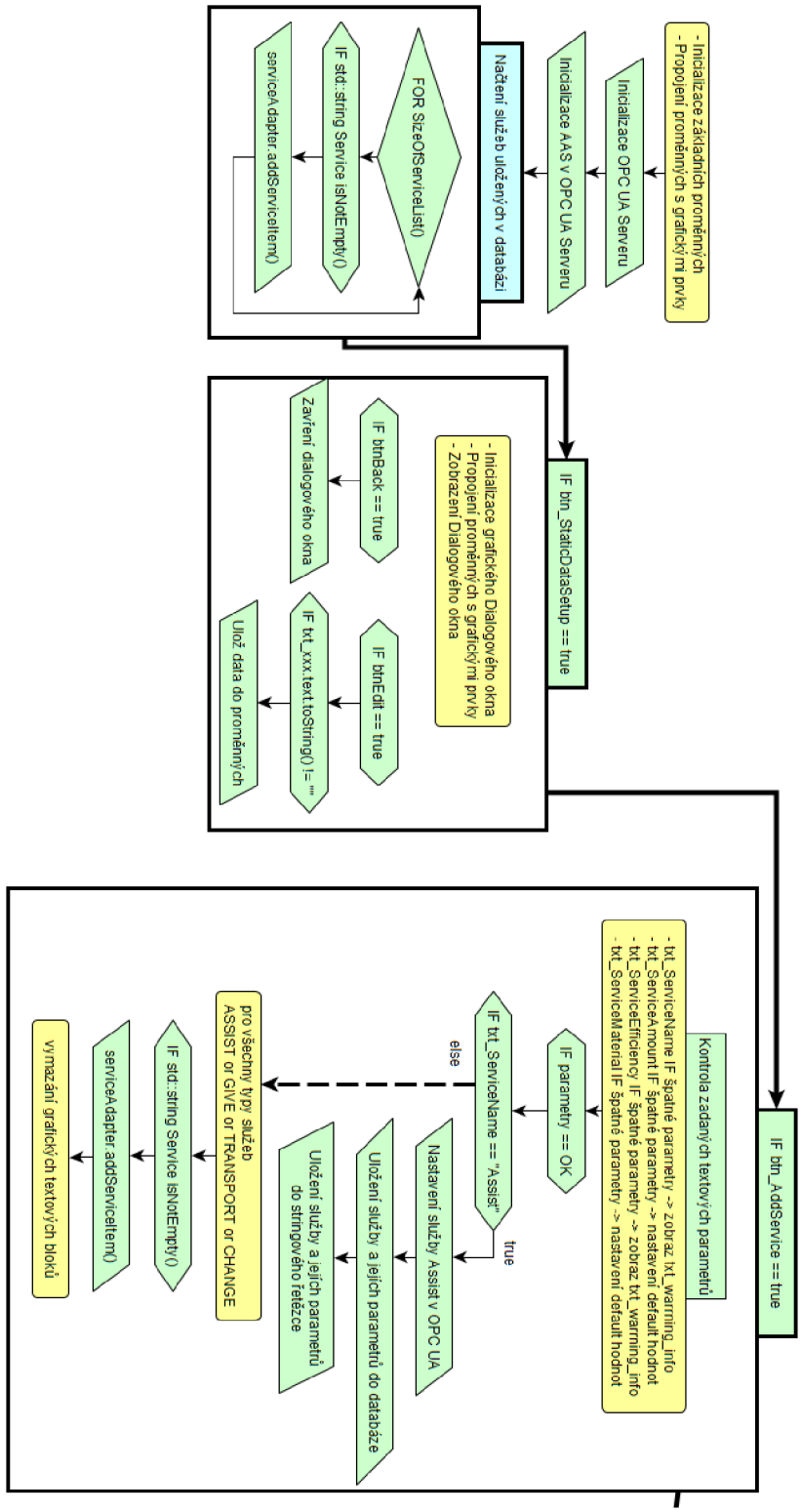


Obr. A.1: Diagram Kotlin kódu, pracovní aktivita/obrazovka hlavní vlákno

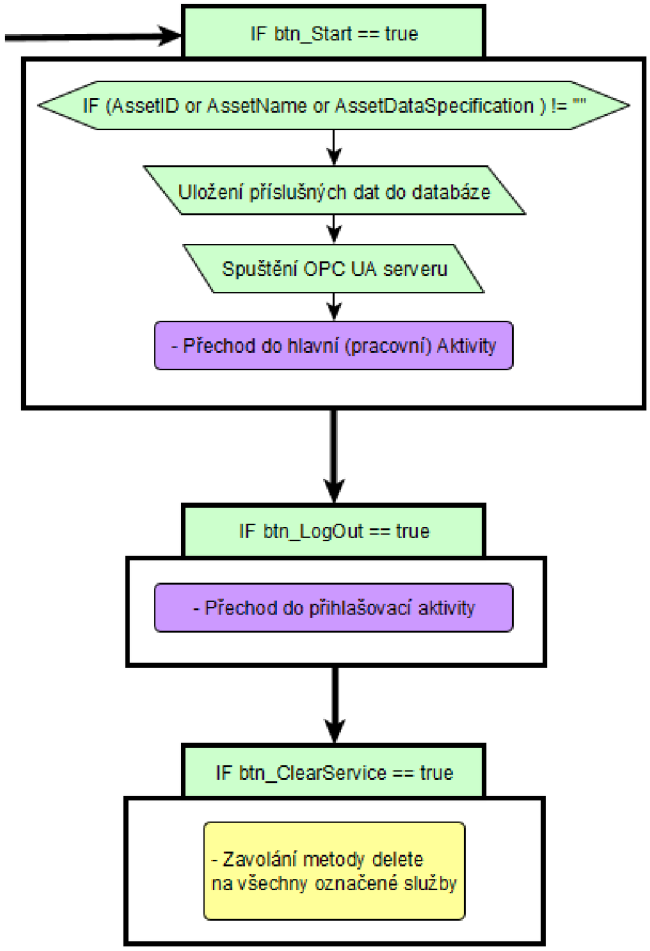


Obr. A.2: Diagram Kotlin kódu, hlavní pracovní obrazovka druhé vlákno

A.2 Aktivita/obrazovka s nastavováním parametrů

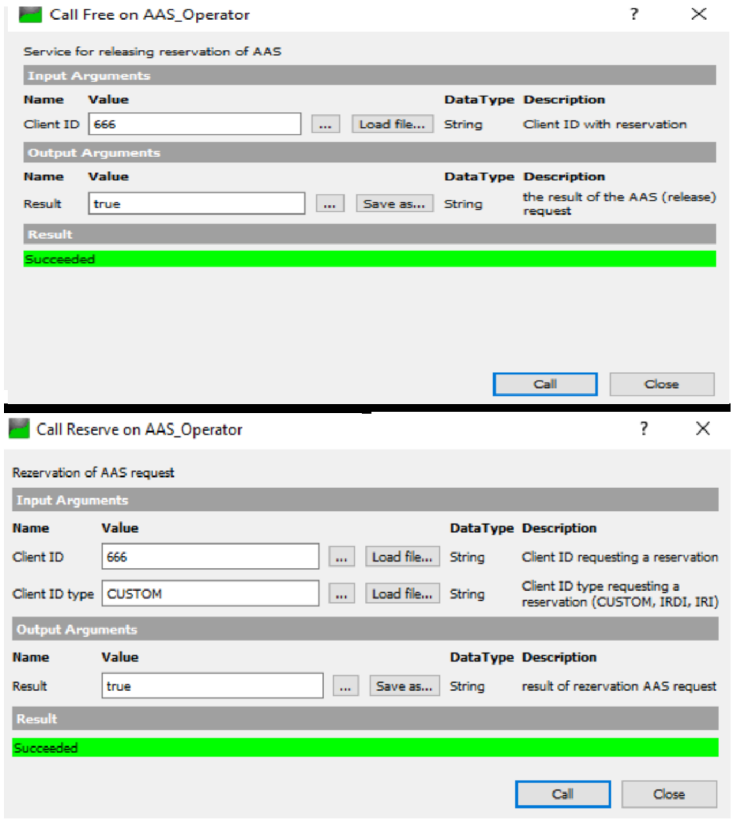


Obr. A.3: Diagram Kotlin kódu, aktivita/obrazovka s nastavováním parametrů hlavní vlákno část 1

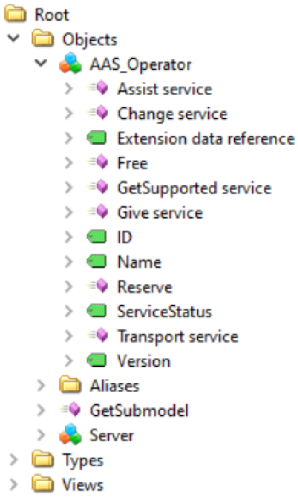


Obr. A.4: Diagram Kotlin kódu, Obrazovka s nastavováním parametrů hlavní vlákno část 2

B Testování komunikace pomocí programu UaExpert



Obr. B.1: Příklad výstupu testů služby Reserve a Free v prostředí UaExpert



Obr. B.2: Příklad navázané komunikace programu UaExpert s AAS

C Obsah na přiloženém DVD

C.1 Projekt mobilní aplikace v prostředí Android studio

C.2 Mobilní aplikace v podobě .apk

C.3 Návod pro obsluhu mobilní aplikace AAS .docx

C.4 Návod pro obsluhu mobilní aplikace AAS .pdf