ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

NÁVRH SPOLEČNÉHO POPISU SLUŽEB PRŮMYSLOVÉHO ŘÍDICÍHO SYSTÉMU A PODNIKOVÉHO INFORMAČNÍHO SYSTÉMU

Bc. Jakub ŠTOLA

DIPLOMOVÁ PRÁCE
2009

Vedoucí práce: Ing. Jan Bezdíček, Ph.D.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne 23.1.2009 …….

podpis

Poděkování

Na tomto místě bych rád poděkoval především vedoucímu mé práce Ing. Janu Bezdíčkovi, Ph.D. a jeho kolegovi Ing. Ladislavu Bumbálkovi za jejich odborné konzultace a připomínky k mé práci, které mě vždy posunuly o krok dále a rozšířily mé poznání dané problematiky. Dále chci poděkovat své rodině, která mi vytváří pohodlné a klidné zázemí po celou dobu studia.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jakub Štola**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Návrh společného popisu služeb průmyslového řídicího systému a podnikového informačního systému**

Pokyny pro vypracování:

1.Prostudovat literaturu zaměřenou na popis služeb informačních systémů(zaměřit se zejména na WSDL).
2.Prostudovat popis některého otevřeného standardu průmyslové komunikace (například CIP) používaného pro sdílení služeb automatizovaného systému řízení.
3.Vybrat vhodné řešení společného popisu služeb průmyslového řídicího systému a podnikového informačního systému. Pokud bude nalezeno více variant, tak provést srovnání.
4.Demonstrovat zvolený popis na několika vybraných službách průmyslového řídicího systému a podnikového informačního systému.
5.Zhodnotit navržené řešení z pohledu reálné použitelnosti.
6.Nepovinné - Vytvořit nástroj (program pro PC), který by po zadání popisu služby řídicího systému či informačního systému v doporučeném jednotném formátu tuto službu zavolal a zobrazil výsledek.

Seznam odborné literatury:

WSDL (http://www.w3.org/TR/wsdl20-primer/), CORBA IDL,
MIDL (http://msdn2.microsoft.com/en-us/library/aa367091.aspx),
OSIDs (http://sourceforge.net/projects/okiproject),
USDL (http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10245/32665/01530890.pdf?temp=x),
UDDI (http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm),
CIP Specification (http://www.odva.org)

Vedoucí: Ing. Jan Bezdíček, Ph.D.

Platnost zadání: do konce zimního semestru 2009/10

prof. Ing. Michael Šebek, DrCs.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 8. 1. 2009

Anotace

Cílem této práce je navrhnout společný popis služeb průmyslového řídicího systému a podnikového informačního systému. Motivace k této práci je uvedena v první kapitole. V následujících dvou kapitolách jsou popsány použité standardy a jejich modely služeb. Čtvrtá kapitola popisuje návrhy možných mapování a společných popisů těchto standardů tak, aby vznikl jednotný popis služeb obou standardů. Na konkrétních příkladech jsou v poslední kapitole demonstrovány možnosti použití společného popisu modelů služeb.

Anotation

The aim of this diploma thesis is a design of a common service description language for industrial control system and enterprise information system. Motivation part for this work is presented in the first chapter. Applied standards and their service models are described in the following two chapters of this thesis. The fourth chapter describes designs of mapping and common description of these standards in a way to create unified description for services of both standards. Concrete examples using the common description are presented in the last chapter of this thesis.

Obsah

Seznam obrázků ..... **iii**

Seznam tabulek ..... **iii**

1 Úvod **1**

2 Popis služeb podnikového informačního systému **3**

2.1 Popis WSDL *4*
2.1.1 Koncepční model WSDL 2.0 5

2.2 UML popis modelu služeb podnikového informačního systému *6*

2.3 Příklad použití WSDL *8*
2.3.1 Konkrétní příklad WSDL dokumentu 10

3 Popis služeb průmyslového řídicího systému **12**

3.1 Popis CIP *13*

3.2 UML model služeb průmyslového řídicího systému *15*

3.3 Příklad popisu CIP objektu a jeho služeb *17*

3.4 Konkrétní popis vybraného CIP objektu *19*
3.4.1 Atributy Identifikačního objektu 19
3.4.2 Služby Identifikačního objektu 21
3.4.2.1 Služba Get_Attribute_Single 21
3.4.2.2 Služba Reset 22
3.4.2.3 Služba Get_Attribute_All 22
3.4.2.4 Služba Set_Attribute_Single 23
3.4.2.5 Služba Find_Next_Object_Instance 24
3.4.2.6 Služba Get_Member 24

4 Společný popis služeb řídicího a informačního systému **25**

4.1 Mapování modelů služeb *25*

4.2 Mapování datových typů *28*

5 Příklady společného popisu **30**

5.1 Konkrétní realizace WSDL dokumentu *30*
5.1.1 WSDL popis služeb Identifikačního objektu – Varianta I 30
5.1.2 WSDL popis služeb Identifikačního objektu – Varianta II 39

5.2 Návrh komunikace mezi systémy *46*

6 Závěr **48**

Literatura a odkazy **50**

Příloha A – Stručný popis UML **52**

Příloha B – Obsah přiloženého CD **57**

Seznam obrázků

Obrázek 1: Jednotlivé vrstvy automatizovaného výrobního procesu [25] 2
Obrázek 2: Dilema podnikového IT [26] 4
Obrázek 3: Koncepční model WSDL 2.0 [27] 6
Obrázek 4: UML popis WSDL modelu služeb 7
Obrázek 5: Základní myšlenka síťových služeb[28] 8
Obrázek 6: Příklad komunikace pomocí síťových služeb[28] 9
Obrázek 7: Ukázka služby 10
Obrázek 8: WSDL popis služby 11
Obrázek 9: Komunikační síť založená na protokolu CIP 13
Obrázek 10: Rodina protokolů specifikace CIP [2] 14
Obrázek 11: UML model služeb průmyslového řídicího systému CIP 16
Obrázek 12: Popis specifikace objektu [2] 17
Obrázek 13: Varianty mapování 26
Obrázek 14: Mapování modelů služeb 27
Obrázek 15: Varianta komunikace přes CIP protokol 46
Obrázek 16: Varianta komunikace s komunikačním mostem mezi protokoly HTTP a CIP ... 47

Seznam tabulek

Tabulka 1: Popis atributu [2] 18
Tabulka 2: Popis služby [2] 19
Tabulka 3: Identifikační objekt - atributy třídy 19
Tabulka 4: Identifikační objekt - atributy instance 20
Tabulka 5: Identifikační objekt - společné služby 21
Tabulka 6: Data přijímaná službou Get_Attribute_Single jako žádost 21
Tabulka 7: Data odesílaná službou Get_Attribute_Single jako úspěšná odpověď 22
Tabulka 8: Data přijímaná službou Reset jako žádost 22
Tabulka 9: Data odesílaná službou Get_Attribute_All jako úspěšná odpověď 23
Tabulka 10: Data přijímaná službou Set_Attribute_Single jako žádost 23
Tabulka 11: Data přijímaná službou Find_Next_Object_Instance jako žádost 24
Tabulka 12: Data odesílaná službou Find_Next_Object_Instance jako úspěšná odpověď 24
Tabulka 13: Data přijímaná službou Get_Member jako žádost 24
Tabulka 14: Data odesílaná službou Get_Member jako úspěšná odpověď 24
Tabulka 15: Mapování datových typů 29

1 Úvod

V dnešní době velmi rozvinuté automatizace a počítačového zpracovaní informací na všech úrovních podnikové infrastruktury existuje mnoho standardů a norem používaných v informačních systémech a v systémech průmyslové automatizace. Se stále se zvyšujícím tlakem na co nejaktuálnější přístup k informacím roste i požadavek na propojení všech systémů podnikové infrastruktury do jednoho celku. Cílem této diplomové práce je navrhnout a vyřešit společný popis služeb podnikového informačního systému a služeb průmyslového řídicího systému. Motivací pro tuto práci je možnost snadného napojení těchto dvou domén výrobního procesu a tím i usnadnění komunikace z nejvyšších úrovní řízení, tzv. manažerských informačních systémů, až po nejnižší úroveň, jímž je již například konkrétní programovatelný automat nebo senzor na výrobní lince.

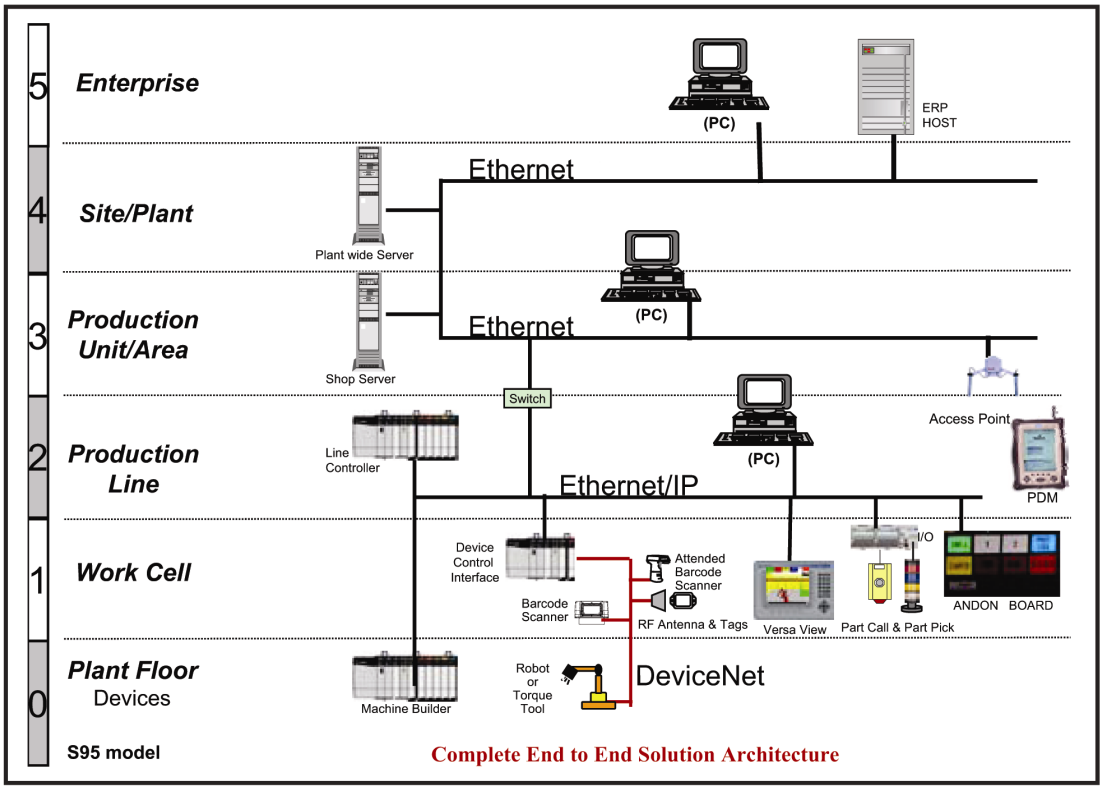
Spodní vrstva je tvořena průmyslovým řídicím systémem. Průmyslový řídicí systém typicky používá pro komunikaci (sdílení dat, vzájemnou interoperabilitu, atd.) specifické protokoly a sítě umožňující řízení v reálném čase. Většina těchto protokolů je dnes již otevřená, tzn. umožňuje i napojení na další systémy.

Požadavky na výrobu a další manažerské informace se nachází na nejvyšší úrovni a jednotlivé složky spolu komunikují přes daný podnikový informační systém. Komunikace pracuje konkrétně přes jiné typy protokolů, které jsou, na rozdíl od těch používaných v řídicím systému, zaměřeny typicky na přenos větších objemů dat a to většinou v podobě "dávek" (zpracování objednávek, plán výroby, přehled výroby za minulý měsíc, atd.). Navíc na této úrovni v drtivé většině případů odpadá požadavek na přenos dat v reálném a přesně deterministickém čase.

Zde se nachází problém, jak tyto dvě oddělené části propojit tak, aby bylo možné celý výrobní proces integrovat do jednoho celku. Takováto integrace by s sebou přinesla celou řadu výhod. Tou největší je možnost jednotného on-line přístupu k datům a informacím v celém podnikovém systému.

Podnikový systém využívá pro komunikaci například konceptu tzv. síťových služeb. Síťová služba je programová složka nezávislá na platformě a implementaci. Služba může být popsána pomocí jazyka popisu služeb, zveřejněna v registru služeb, vyvolána pomocí příslušného aplikačního rozhraní a spojována s ostatními službami. Na tento koncept je zaměřena i tato práce.

Na *Obrázku 1* jsou znázorněny jednotlivé vrstvy automatizovaného výrobního procesu, přičemž vrstvy 0-2 představují průmyslový řídicí systém a vrstvy 3-5 podnikový informační systém.



Obrázek 1: Jednotlivé vrstvy automatizovaného výrobního procesu [25]

Tato práce si tedy dává za úkol nalézt společný popis služeb podnikového a průmyslového systému tak, aby tyto dvě oblasti bylo možné snadněji zakomponovat do jednoho celku. Služba průmyslového řídícího systému by se tedy měla chovat stejně a být přístupná jako jakákoliv další služba podnikového systému. Tímto by se dosáhlo provázanosti těchto dvou vrstev výrobního procesu.

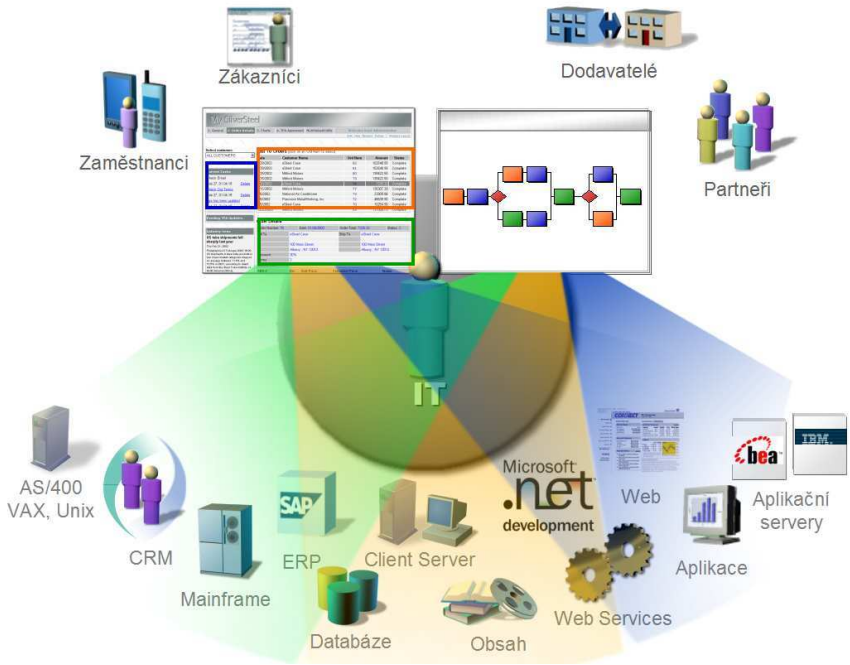
2 Popis služeb podnikového informačního systému

V době stále se rozvíjejících a nově vznikajících heterogenních aplikací na úrovni podnikového managementu (viz. *Obrázek 2*) je klíčová snaha o sjednocenou správu a provoz těchto aplikací. Za tímto účelem bylo vytvořeno mnoho standardů jako např. CORBA [5], COM+ [6], DCE [7], MIDL [8] apod. Hlavním nedostatkem výše jmenovaných standardů je to, že nejsou plně otevřenými a tudíž snadno upravitelnými a všeobecně použitelnými. Standardy sice byly vytvářeny jako otevřené, ale postupným vývojem tuto otevřenost ztratily. Typickým příkladem je standard CORBA. Konsorcium firem, které tento standard vytvářelo, nebylo schopno dostatečně pružně řešit vzniklé problémy. Začali tak vznikat velké problémy s interoperabilitou mezi různými implementacemi CORBA.

Dalším z nedostatků výše jmenovaných standardů je, že využívají zejména konceptu vzdáleného volání procedur (*angl. Remote Procedure Call [9], dále jen RPC*). Hlavní nedostatek RPC je v tom, že je principielně založen na blokujícím volání (tedy synchronní volání vzdálené procedury), kdy klient čeká na dokončení volané vzdálené procedury než začne vykonávat nějakou další činnost. Největší komplikace tedy nastávají, pokud odpověď z nějakého důvodu vůbec nedorazí. K tomuto jevu může dojít v případě poruchy. Dalším důvodem mohou být například firewally různých systémů, které tento požadavek nepropustí. Systému, který čeká na odpověď, hrozí v tomto případě zamrznutí nebo jiné narušení běžné činnosti.

Výše zmíněné nedostatky se podařilo částečně odstranit vytvořením konceptu síťových služeb pro jehož popis byl navržen jako otevřený standard jazyk WSDL [1], a který principielně staví na asynchronním volání vzdálené služby. Takto navržený koncept umožňuje komunikovat po internetu, například přes obecný HTTP [10] protokol a pro výměnu zpráv

využívat protokol SOAP [11]. Tímto koncept získává nezávislost na platformě a je všestranně použitelný. Z tohoto důvodu byl standard WSDL vybrán též pro tuto práci a cílem bude tento standard popisu síťových služeb rozšířit i na úroveň průmyslového řídicího sytému.



Obrázek 2: Dilema podnikového IT [26]

2.1 Popis WSDL

Jazyk pro popis síťových služeb (*angl. Web Services Description Language [1], dále jen WSDL*) je jazyk založený na formátu XML [3], který definuje a popisuje model síťových služeb. Se standardizací komunikačních protokolů a formátů zpráv vyvstává stále více potřeba možnosti popisu komunikace nějakým strukturovaným způsobem. WSDL definuje služby jako množinu síťových koncových bodů. Abstraktní definice koncových bodů a zpráv jsou oddělené od jejich konkrétních instancí, což umožňuje opakované použití těchto definic. Koncový bod je definován síťovou adresou a konkrétní vazbou (*angl. Binding*), určující typ protokolu a kódování, které se používá při komunikaci s konkrétní instancí dané služby. Množina těchto koncových bodů definuje službu. Zprávy jsou abstraktním popisem dat, která mají být přenášena. Rozhraní je definováno jako abstraktní soubor podporovaných operací. Konkrétní protokol a specifikace datových formátů pro příslušné rozhraní tvoří vazbu (*angl. Binding*), podle níž jsou operace a zprávy svázány s konkrétním síťovým

protokolem a formátem zpráv. Tímto způsobem popisuje WSDL veřejné rozhraní síťové služby.

WSDL je dnes široce akceptovaný standard, který existuje v několika verzích – WSDL 1.0, WSDL 1.1 a WSDL 2.0. WSDL 1.1 se stal plně akceptovaným de facto standardem v IT světě. Většina výrobců softwaru nyní podporuje přístup k aplikacím právě přes síťové služby (Oracle [17], SAP [18], aj.) popsané pomocí verze WSDL 1.1. WSDL 2.0 je současnou nejnovější verzí tohoto standardu a je plně podporován a doporučován konsorciem webových standardů (*angl. The World Wide Web Consortium - W3C*) [4]. Tato nejnovější verze přináší mnoho důležitých rozšíření a modifikací předchozích verzí pro ještě flexibilnější popis služeb:

- Standard umožňuje daleko větší rozšiřitelnost základních konceptů
- Definice zpráv a chyb je možné sdílet pro více než jednu službu
- Standard definuje pojem rozhraní (*angl. Interface*). Rozhraní definuje operaci, nebo skupinu operací, které spolu logicky souvisejí.
- Rozhraní mohou být rozšířena pomocí dědičnosti

Právě pro své vlastnosti je WSDL 2.0 použit i v této práci.

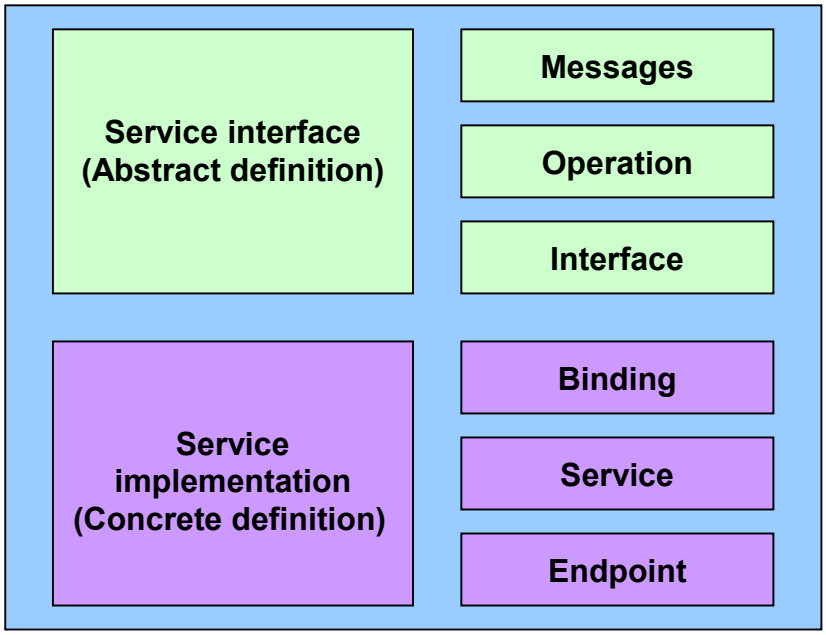
2.1.1 Koncepční model WSDL 2.0

Popis síťové služby může být rozdělen do dvou částí. Koncepční model WSDL je zobrazen na *Obrázku 3*.

V *abstraktní části* WSDL popisuje službu z hlediska zpráv, které daná služba přijímá, a které posílá. Popis zpráv je typicky pomocí XML schématu [30]. Typ výměny zpráv (*angl. Message Exchange Pattern*) definuje posloupnost a význam obsahu jednotlivých zpráv. Pro každou operaci (*angl. Operation*) se definuje konkrétní typ výměny zpráv. Rozhraní (*angl. Interface*) slučuje tyto operace tak, aby byly nezávislé na transportní a fyzické vrstvě.

V *konkrétní části* blok nazvaný vazba (*angl. Binding*) specifikuje transportní a fyzický formát pro dané abstraktní rozhraní. Koncový bod (*angl. Endpoint*) definuje síťové adresy pro danou vazbu. Poslední částí konkrétního popisu je služba (*angl. Service*). Služba popisuje koncové body, které implementují rozhraní a definují adresu, kde se tato služba v systému nachází. V konkrétní části WSDL je tedy určeno především komunikační rozhraní přes které

je dané služby možné volat a adresa koncového bodu. Obsahuje tedy konkrétní protokoly a vazby.



Obrázek 3: Koncepční model WSDL 2.0 [27]

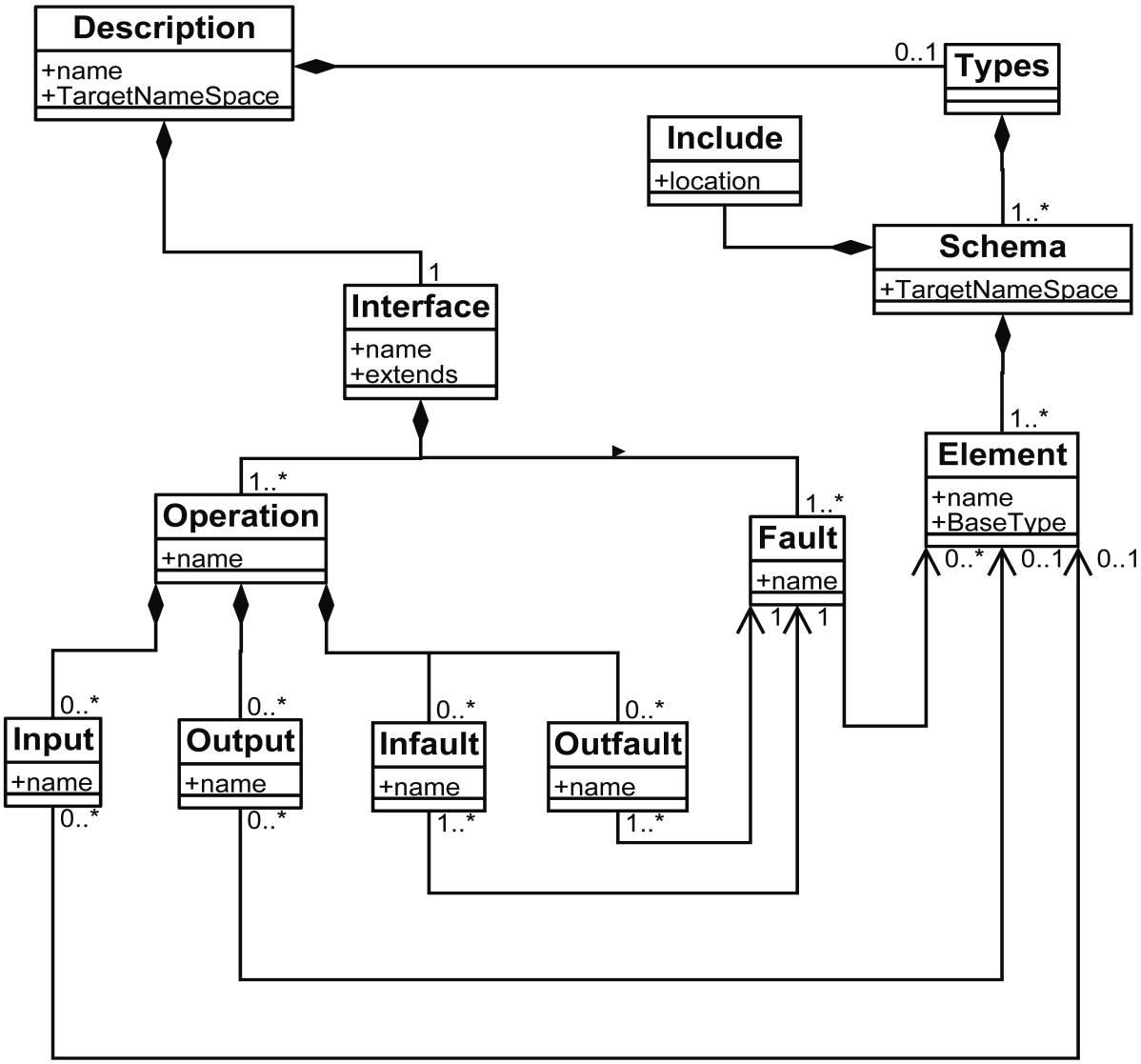
Je třeba poznamenat, že tato práce se zaměřuje především na mapování obou standardů (podnikového a průmyslového) na jejich abstraktní úrovni. Nicméně pro praktickou realizaci a konkrétní ukázku propojení obou standardů je nutné nadefinovat i tuto konkrétní část WSDL - specifikaci protokolu a kódování označovanou ve WSDL standardu jako vazba (*angl. Binding*).

2.2 UML popis modelu služeb podnikového informačního systému

Obrázek 4 popisuje WSDL 2.0 model pomocí UML [13] diagramu. UML popis je z důvodu přehlednosti a všeobecné srozumitelnosti s popisy jednotlivých prvků v anglickém jazyce. V následujícím popisu modelu jsou použity české ekvivalenty s odkazy na dané prvky v anglickém jazyce. Model služeb představuje pouze abstraktní část WSDL popisu. Konkrétní částí WSDL popisu se v této kapitole práce nezabývá.

Základním prvkem modelu je blok popisu (*angl. Description*), který tvoří obálku každého WSDL dokumentu. Jeho atributy jsou název (*angl. Name*) a cílový jmenný prostor (*angl. TargetNameSpace*). Jmenné prostory se používají k zamezení konfliktů mezi jmény pokud je integrováno více služeb.

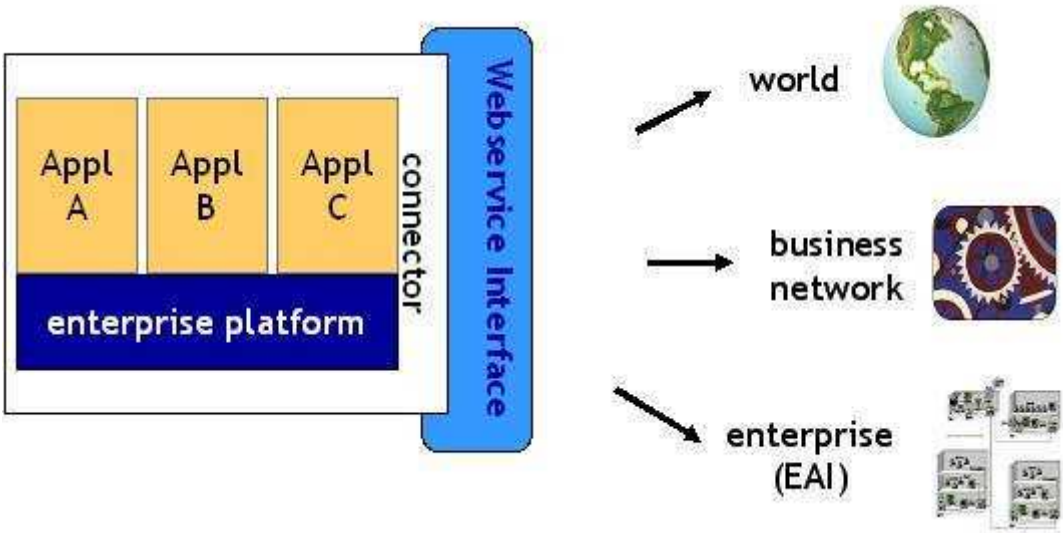
Blok popisu obsahuje následující části: rozhraní (*angl. Interface*) a typy (*angl. Types*). Rozhraní je klíčovou částí WSDL 2.0 modelu. Popisuje totiž operace (*angl. Operation*), které daná služba nabízí a jejich chybová hlášení (*angl. Fault*). Operace sjednocuje množinu zpráv, které si vyměňují poskytovatel služby a její uživatel. Každá operace může přijímat vstupní zprávy (*angl. Input*), posílat výstupní zprávy (*angl. Output*) a případně chybové zprávy (*angl. Infault a Outfault*) pokud je jich zapotřebí. Každá zpráva má svůj název (*angl. Name*) a konkrétní data (*angl. Element*). Data musí být podporovaného typu (*angl. Types*). WSDL využívá datová schémata popsaná v bloku schéma (*angl. Schema*), kde jsou definované jmenné prostory a datové typy (*int, float, string* atd.). WSDL dovoluje použití předem definovaných XML schémat pomocí bloku Vložit (*angl. Include)* s informací o pozici daného dokumentu (*angl. Location*).



Obrázek 4: UML popis WSDL modelu služeb

2.3 Příklad použití WSDL

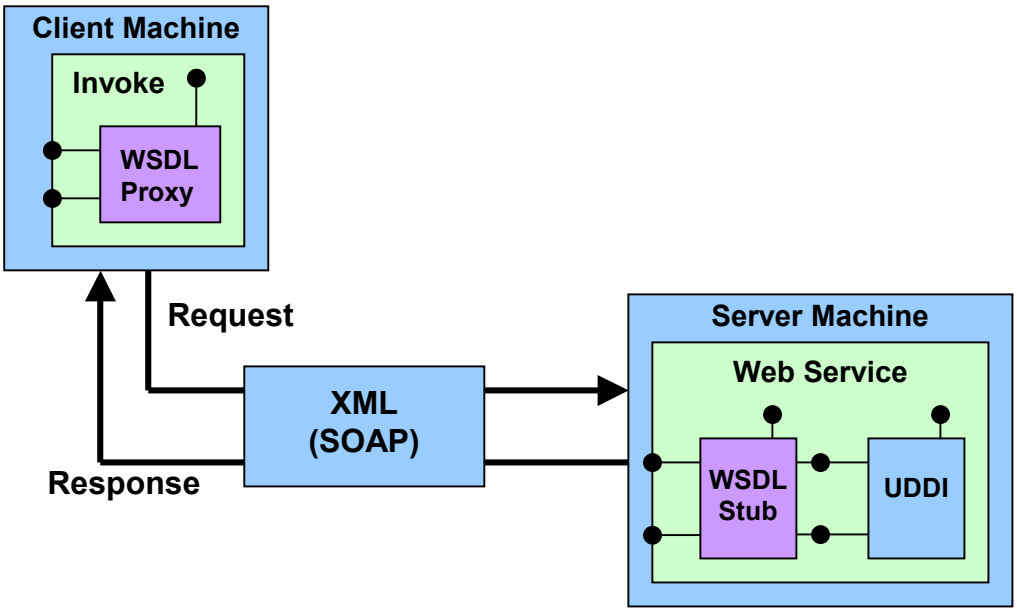
Síťovou službu můžeme zjednodušeně chápat jako libovolný požadavek ve smyslu žádost-odpověď. V rámci průmyslové automatizace může představovat takovou službu například požadavek na výrobu nového automobilu. Klientem je v tomto případě např. plánovací aplikace na úrovni podnikového informačního systému výrobního závodu a koncovým poskytovatelem služby je příslušný řídicí systém na výrobní lince. Na *Obrázku 5* je znázorněn princip této myšlenky. Rozhraní síťové služby (*angl. Web Service Interface*) může mít v našem případě například podobu rozhraní HTTP-SOAP a popis dané služby je definován a popsán pomocí WSDL dokumentu.



Obrázek 5: Základní myšlenka síťových služeb[28]

Ostatní aplikace komunikují s konkrétní službou pomocí protokolu definovaného WSDL dokumentem (nejčastěji, jako v níže uvedeném příkladu, pomoci SOAP zpráv přenášených po HTTP protokolu), jak je znázorněno na *Obrázku 6.*

Dnes existuje celá řada vývojových prostředí, které umožňují pohodlně danou síťovou službu zavolat a otestovat na základě znalosti WSDL dokumentu. Každou aplikaci je tudíž nutné opatřit příslušnou obálkou (*angl. Wrapper*), která je dnes již součástí standardní výbavy většiny vývojových prostředí (např. Visual Studio [19], NetBeans [20], Eclipse [21] aj.). S pomocí této obálky se pak konkrétní služba navenek chová jako tzv. černá skřínka s danými vstupy a výstupy, kterou lze volat jako libovolnou funkci nebo metodu.

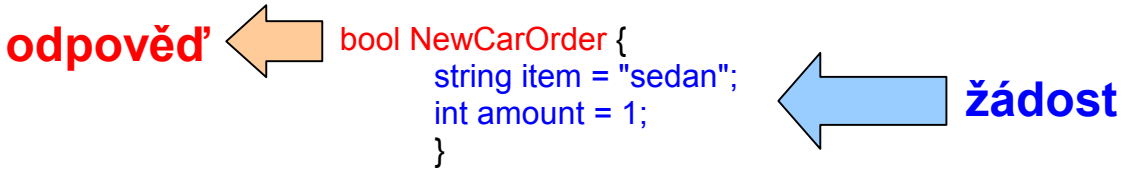


Obrázek 6: Příklad komunikace pomocí síťových služeb[28]

2.3.1 Konkrétní příklad WSDL dokumentu

V této kapitole bude na konkrétním příkladu předvedeno, jak lze sestavit WSDL popis jednoduché služby.

Příkladem takové služby může být např. již zmíněná žádost o výrobu nového automobilu. *Obrázek 7* ukazuje takovou službu jako metodu napsanou v jazyce C#:



Obrázek 7: Ukázka služby

Žádostí je v tomto případě typ a počet kusů objednávaného automobilu a odpovědí je potvrzení přijetí požadavku ve formátu logické boolovské hodnoty. V rámci zjednodušení tato služba negeneruje žádná chybová hlášení.

Tato služba může být popsána pomocí následujícího WSDL dokumentu. Pro lepší názornost je příklad WSDL dokumentu graficky rozdělen do následujících klíčových částí, kterými jsou:

Description	základní část každého WSDL dokumentu, definuje např. jmenný prostor
Types	část obsahující definice používaných datových typů (např. XSD)
Interface	abstraktní definice operací a zpráv dané služby
Binding	specifikace transportního a fyzického formátu pro dané abstraktní rozhraní
Service	definice koncových bodů implementující společné rozhraní

Kompletní WSDL dokument pro popis této služby je na následující stránce (viz. *Obrázek 8*).

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
    xmlns:wsdl="http://www.w3.org/ns/wsdl"
    targetNamespace="urn:wsdlSamples"
    xmlns:tns="urn:wsdlSamples"
    xmlns:dtns="urn:wsdlSamplesData">
```

```
<wsdl:types>
      <xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
      targetNamespace="urn:wsdlSamplesData">

            <xs:complexType name="NewCarOrderRequest">
                <xs:sequence>
                    <xs:element name="item" type="xs:string"/>
                    <xs:element name="amount" type="xs:integer"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name=" NewCarOrderResponse">
                <xs:sequence>
                    <xs:element name="OrderStatus" type="xs:boolean"/>
                </xs:sequence>
            </xs:complexType>
      </xs:schema>
</wsdl:types>
```

```
<wsdl:interface name="NewCarOrderInterface">

      <wsdl:operation name="NewCarOrder"
            pattern="http://www.w3.org/ns/wsdl/in-out">
                  <wsdl:input messageLabel="In"
                        element="dtns: NewCarOrderRequest "/>
                  <wsdl:output messageLabel="Out"
                        element="dtns: NewCarOrderResponse "/>
      </wsdl:operation>
</wsdl:interface>
```

```
<wsdl:binding name="TCPBinding"
             interface="tns: NewCarOrderInterface">
        <wsdl:operation ref="tns:NewCarOrder"/>
    </wsdl:binding>
```

```
<wsdl:service name="NewCarOrder"
        interface="tns: NewCarOrderInterface ">
        <wsdl:endpoint name="NewCarOrderEndPoint"
            binding="tns:TCPBinding"
            address="tcp://localhost/NewCarOrder"/>
    </wsdl:service>
```

```
</wsdl:description>
```

Obrázek 8: WSDL popis služby

3 Popis služeb průmyslového řídicího systému

Stejně jako v případě podnikových informačních systému i v oblasti průmyslových systémů existuje mnoho různorodých standardů komunikačních sítí a protokolů (např. Profibus [14], Interbus [15], CANopen [16]). Z tohoto důvodu i v této oblasti narážíme na problém standardizace a snahy o snadné sjednocení jednotlivých komponentů řídících systémů. Jednotlivé sítě používané ve výrobních závodech jsou optimalizovány pro specifické aplikace. Z důvodu zvýšení účinnosti, spolehlivosti a zejména ekonomické efektivnosti celého systému jsou vývojáři nuceni zavádět vícero odlišných sítí. V praxi pak existuje ve výrobním závodu velké množství specializovaných a všeobecně nekompatibilních sítí pracujících na jednom společném prostoru.

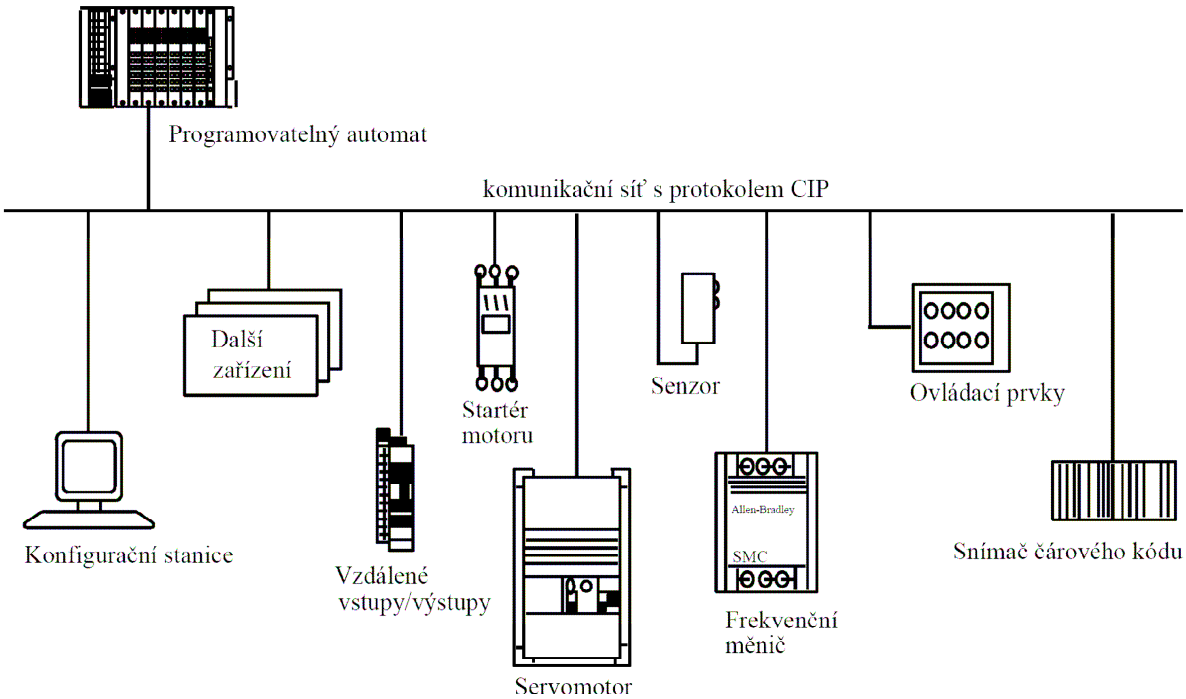
Z těchto důvodů se v minulých letech rapidně zvýšila snaha o nalezení otevřeného standardu pro komunikační sítě, díky kterému by bylo možné jednoduše propojit různorodé částí výrobního systému a procesu. Tyto požadavky splňuje například průmyslový standard CIP (*angl. Common Industrial Protocol - Obecný průmyslový protokol, dále jen CIP*) [2] spravovaný organizací ODVA [31]. Jeho hlavní výhodou je, že se jedná o otevřený standard a je tudíž snadné ho přizpůsobit pro danou konkrétní aplikaci průmyslového řídicího systému. Tato práce se zabývá právě tímto standardem a jeho možnou integrací s výše popsaným standardem podnikového informačního systému.

Cílem je tedy nalézt analogie mezi popisem služeb definovaných standardem CIP a popisem služeb dle standardu WSDL a tím navrhnout možný jednotný popis služeb na všech úrovních podnikové infrastruktury. Navržené řešení lze aplikovat nejen na sjednocení popisu služeb podnikového informačního systému s popisem služeb průmyslového řídicího systému

využívajícího protokol CIP, ale i na integraci ostatních standardů využívaných v průmyslové automatizaci.

3.1 Popis CIP

CIP je otevřený objektově orientovaný protokol pro aplikace průmyslové automatizace. Protokol zahrnuje komplexní sadu služeb a zpráv pro různorodé využití v oblastech jako je automobilový, chemický, či potravinářský průmysl. CIP zajišťuje spojení mezi průmyslovými zařízeními, jako jsou například senzory a regulátory, a zařízeními vyšší úrovně – programovatelnými automaty a panely operátorů (viz. *Obrázek 9*). Abstraktní popis pomocí CIP specifikace je nezávislý na fyzické a linkové vrstvě, zahrnuje v sobě relační, prezentační a aplikační vrstvu ISO-OSI [29] modelu. CIP umožňuje jak konfiguraci zařízení, tak přístup k datům již v běžícím systému, což výrazně snižuje čas návrhu a instalace a tím náklady na zavedení a údržbu takového systému. Na *Obrázku 10* je znázorněna rodina protokolů specifikace CIP. Jednotlivé specifické protokoly pracují na různých přenosových mediích (CAN [22], Ethernet [23]).

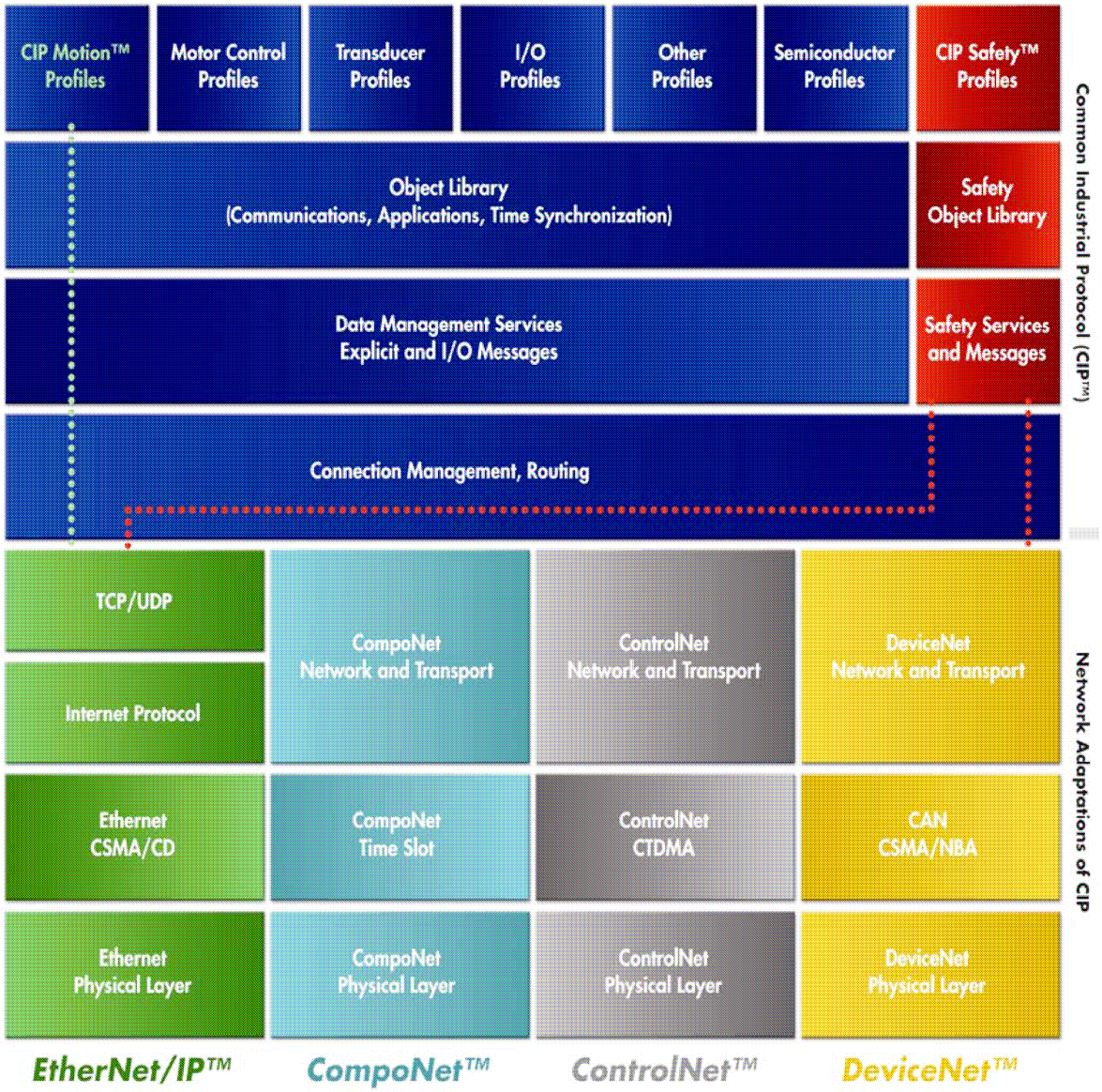


Obrázek 9: Komunikační síť založená na protokolu CIP

Hlavními úkoly protokolu specifikace CIP jsou:

- přenos časově kritických dat mezi I/O zařízeními a inteligentními prvky řídicí aplikace
- přenos ostatních, časově nekritických dat jako například konfiguračních a diagnostických dat

Každé zařízení připojené k síti, které využívá protokol CIP, se skládá z objektů poskytujících specifické služby. První skupinu tvoří objekty, které se povinně vyskytují u všech zařízení. Druhou skupinu tvoří objekty, jejichž výskyt je závislý na konkrétní realizaci zařízení. Třetí skupinou jsou objekty specifické pro danou síť (DeviceNet, ControlNet, EtherNet/IP).



Obrázek 10: Rodina protokolů specifikace CIP [2]

Specifikace normy CIP byla vytvořena v 80. letech a tudíž její popis ještě zcela nesplňuje dnešní běžně používaný standard objektově-orientovaného popisu. Narazíme zde tudíž na drobné odchylky v terminologii. Například význam termínů třída a instance je odlišný od dnes běžně zažitého chápání těchto termínů. Termín označený v normě CIP jako třída může být z dnešního pohledu chápáno spíše jako kolekce objektů, které jsou v normě CIP popsány jako instance (*angl. Instance*).

3.2 UML model služeb průmyslového řídicího systému

Aby bylo možné provést srovnání a navrhnout mapování mezi popisem služeb definovaných standardy CIP a WSDL, tak i pro výše popsaný průmyslový řídicí systém standardu CIP byl navržen jeho UML model služeb (viz. *Obrázek 11*). UML popis je z důvodu přehlednosti a všeobecné srozumitelnosti s popisy jednotlivých prvků opět v anglickém jazyce. V následujícím popisu modelu jsou opět použity české ekvivalenty s odkazy na dané prvky v anglickém jazyce.

Základním prvkem modelu služeb průmyslového řídicího systému je třída (*angl. Class*). Jak již bylo zmíněno výše, třída je v prostředí CIP chápána spíše jako kolekce objektů, popsaných v standardu CIP jako instance (*angl. Instance)*. Jednotlivé třídy mají své identifikační číslo (*angl. Class ID*). Každá třída může vytvořit jednu či více konkrétních instancí daného objektu. Také každá nově vytvořená instance (*Instance*) obdrží své unikátní identifikační číslo (*angl. Instance ID*) v rámci jedné třídy na daném uzlu protokolu CIP.

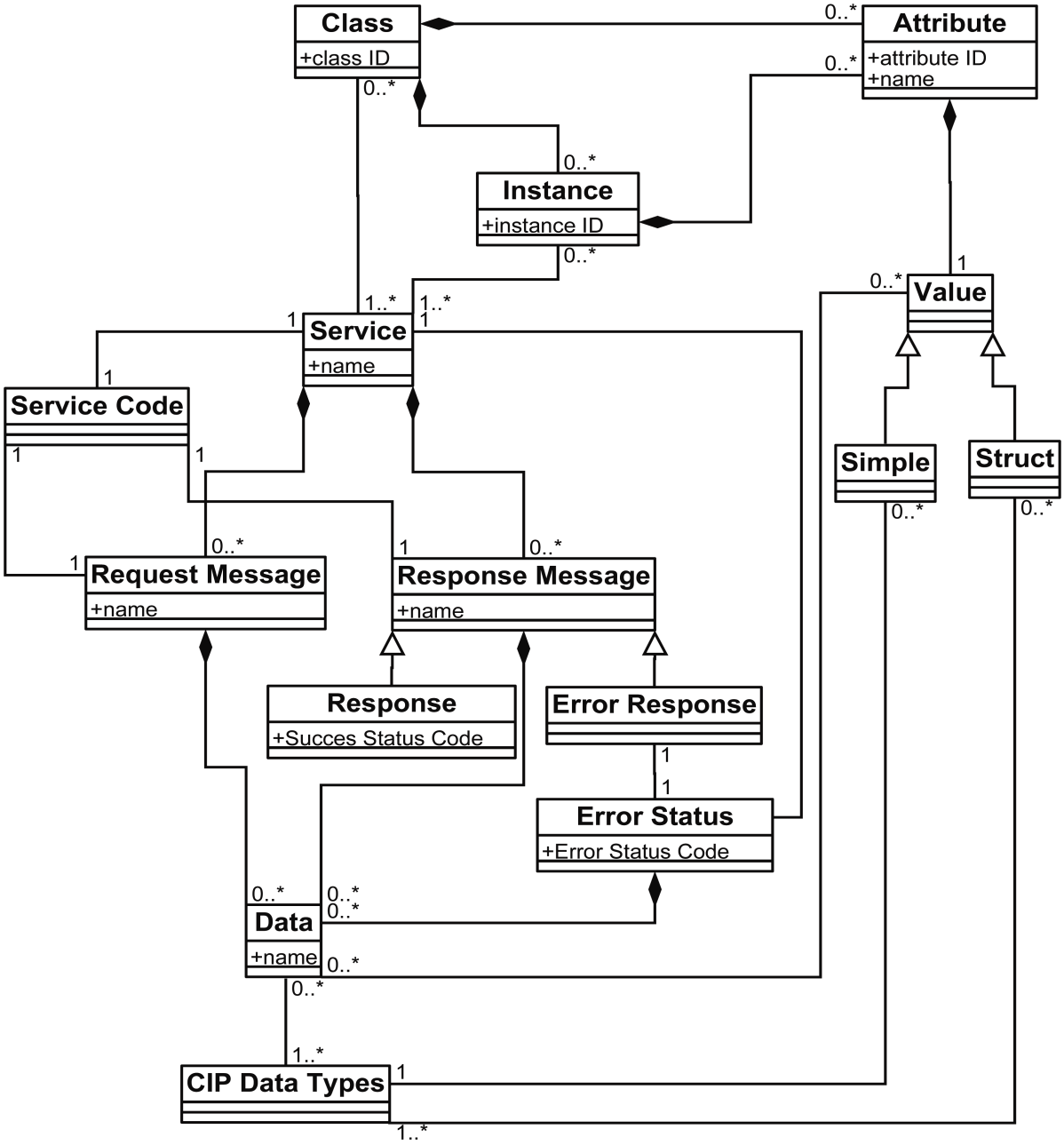
Každá třída a instance mají své atributy (*angl. Attribute*). Atribut má svou hodnotu (*angl. Value*), která může být buď jednoduchého (*angl. Simple*) nebo strukturovaného (*angl. Struct*) typu. Všechny podporované datové typy jsou obsaženy v normě CIP (*angl. CIP Data Types*).

Samotná služba (*angl. Service*) průmyslového řídicího systému CIP je vázána buď k dané třídě nebo k instanci. Každá služba má svůj unikátní kód služby (*angl. Service Code*), což je hexadecimální hodnota. Spolu s identifikačním číslem třídy nebo instance jednoznačně určují, jaká služba se bude volat na jakém objektu.

V tomto modelu rozlišujeme dva základní typy zpráv: žádost (*angl. Service Request Message*) a odpověď (*angl. Service Response Message*). Odpověď je v závislosti na příslušné odezvě systému buď konkrétní odpovědí (*angl. Response*) na danou žádost obsahující potvrzení úspěšného vykonání dané služby (*angl. Success Status Code*) nebo, za předpokladu

nesplnění některých požadavků, chybovým hlášením (*angl. Service Error Response*). Chybové hlášení musí obsahovat status chyby (*angl. Error Status*) s příslušným chybovým kódem (*angl. Error Status Code*).

Veškerá data (*angl. Data*) obsažená v žádosti, odpovědi nebo v případném chybovém hlášení musí samozřejmě opět vyhovovat podporovaným datovým typům standardu CIP (*angl. CIP Data Types*).

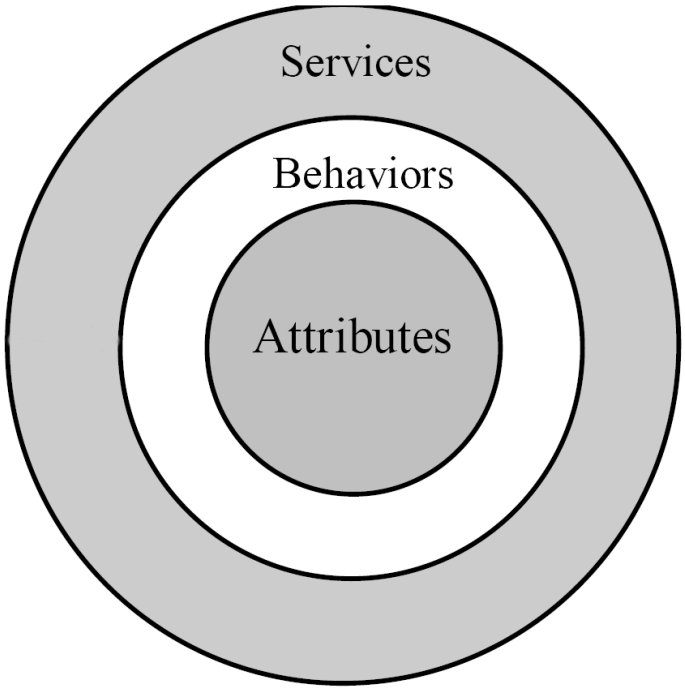


Obrázek 11: UML model služeb průmyslového řídicího systému CIP

3.3 Příklad popisu CIP objektu a jeho služeb

V této kapitole je uvedeno, jakým způsobem jsou objekty ve specifikaci protokolu CIP popsány. Cílem je demonstrovat z jakých částí se každý CIP objekt skládá, jak jsou organizovaná data a především jaké služby daný objekt poskytuje. CIP objekt je abstraktní reprezentace CIP třídy a CIP instance. Standard CIP obsahuje velký soubor obecně definovaných objektů. Tento popis bude dále využit pro mapování služeb protokolu CIP na služby popsané pomocí WSDL dokumentu.

Obrázek 12 představuje základní složky každého CIP objektu. Základními prvky jsou množina příslušných atributů objektu (*angl. Attributes*), chování objektu (*angl. Behaviors*) neboli stavy a události nad daným objektem. CIP objekt je dále charakterizován množinou služeb (*angl. Services*), jednak služeb společných a také specifických pro daný objekt.



Obrázek 12: Popis specifikace objektu [2]

Atributy objektu rozlišujeme na atributy třídy a atributy instance. Každý atributy je definován následující sadou údajů (viz. *Tabulka 1*):

1) identifikační číslo atributu (*angl. Attribute ID*)
2) požadavek na implementaci (*angl. Need in Implemetation*) – nepovinný, povinný a podmíněný
    - *nepovinný parametr* – pokud je parametr definován jako nepovinný, tak musí být definována jeho implicitní hodnota (tovární nastavení)
    - *povinný parametr* – nutný parametr při implementaci objektu
    - *podmíněný parametr* – je nutný pokud je třídou nebo profilem zařízení definované určité chování objektu nebo atributu, který daný parametr vyžaduje
3) přístupová práva (*angl. Access Rule*) – možnost čtení a zápisu
4) indikace obnovování atributu (*angl. NV*) – označení zda hodnota atributu přečká i restart zařízení
5) jméno (*angl. Name*)
6) datový typ (*angl. CIP Data Type*)
7) popis atributu (*angl. Description of Attribute*)
8) význam hodnoty atributu (*angl. Semantics of Values*)

Tabulka 1: Popis atributu [2]

Attribute ID	Need in Implementation	Access Rule	NV	Name	CIP Data Type	Description of Attribute	Semantics of Values
1	2	3	4	5	6	7	8

Jak již bylo zmíněno výše, služby objektu se dělí na společné a specifické pro daný objekt. Služba může být definována pro práci jak na úrovni třídy, tak na úrovni její instance. Každá z těchto služeb je popsána pomocí následujících údajů (viz. *Tabulka 2*):

1) kód služby (*angl. Service Code*)

2,3) požadavek na implementaci (*angl. Need in Implemetation*) na úrovni třídy (*angl. Class level*) a na úrovni instance (*angl. Instance level*)–nepovinný, povinný nebo podmíněný – u každé podmíněné služby musí být definované podmínky, za kterých je služba požadována.

4) jméno služby (*angl. Sevice Name*)
5) popis služby (*angl. Description of Service*)

Tabulka 2: Popis služby [2]

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
1	2	3	4	5

3.4 Konkrétní popis vybraného CIP objektu

Ze specifikace CIP byl pro konkrétní ukázku vybrán Identifikační objekt (*angl. Identity Object*) a jeho služby. Tento objekt je součástí každého produktu pracujícího na protokolu CIP. Objekt poskytuje identifikaci a základní informace o zařízení. Jestliže se zařízení skládá z několika samostatných částí, existuje pro každou část zařízení jedna instance Identifikačního objektu.

3.4.1 Atributy Identifikačního objektu

Identifikační objekt má následující atributy třídy a atributy instance:

Tabulka 3: Identifikační objekt - atributy třídy

ID číslo	požadavek na implementaci	přístupová práva	jméno	CIP datový typ	popis atributu
1	nepovinný	čtení	Revision	UINT	revizní číslo objektu
2	podmíněný	čtení	Max Instance	UINT	maximální počet instancí objektu na úrovni třídy zařízení
3	nepovinný	čtení	Number of Instances	UINT	počet právě vytvořených instancí objektu na úrovni třídy zařízení
4	nepovinný	čtení	Optional Atrribute List	STRUCT	seznam nepovinných atributů užívaných touto implementací třídy
5	nepovinný	čtení	Optional Service List	STRUCT	seznam nepovinných služeb užívaných touto implementací třídy
6	nepovinný	čtení	Maximum ID Number Class Attributes	UINT	ID posledního atributu třídy definice třídy implementované v zařízení
7	nepovinný	čtení	Maximum ID Number Instance Attributes	UINT	ID posledního atributu instance definice třídy implementované v zařízení

Tabulka 4: Identifikační objekt - atributy instance

ID číslo	požadavek na implementaci	přístupová práva	jméno	CIP datový typ	popis atributu
1	povinný	čtení	Vendor ID	UINT	identifikační číslo dodavatele/výrobce
2	povinný	čtení	Device Type	UINT	identifikace typu produktu
3	povinný	čtení	Product Code	UINT	identifikace části produktu nebo jednotlivého dodavatele
4	povinný	čtení	Revision	STRUCT	revize položky, který daný objekt představuje
5	povinný	čtení	Status	WORD	stav zařízení
6	povinný	čtení	Serial Number	UDINT	sériové číslo zařízení
7	povinný	čtení	Product Name	SHORT STRING	jméno produktu
8	nepovinný	čtení	State	USINT	současný stav zařízení reprezentovaný stavovým diagramem
9	nepovinný	čtení	Configuration Consistency Value	UINT	obsah souhlasí s nastavením zařízení
10	nepovinný	čtení/zápis	Heart Beat Interval	USINT	interval udržovacích pulsů
11	nepovinný	zápis	Active Language	STRUCT	aktivní jazyk zařízení
12	nepovinný	čtení	Supported Language List	ARRAY of STRUCT	seznam zařízením podporovaných jazyků
13	nepovinný	čtení	International Product Name	STRINGI	jméno produktu v jiných jazycích
14	nepovinný	zápis	Semaphore	STRUCT	přístup k synchronizačním parametrům
15	nepovinný	zápis	Assigned Name	STRINGI	uživatelské jméno
16	nepovinný	zápis	Assigned Description	STRINGI	uživatelský popis
17	nepovinný	zápis	Geografic Location	STRINGI	geografická poloha uživatele

3.4.2 Služby Identifikačního objektu

Identifikační objekt poskytuje následující společné služby:

Tabulka 5: Identifikační objekt - společné služby

kód služby	požadovaná při implementaci		jméno služby	popis služby
	třída	instance		
0Ehex	podmíněná	nutná	Get_Attribute_Single	vrátí obsah specifikovaného atributu
05hex	volitelná	nutná	Reset	vyvolá reset zařízení
01hex	volitelná	podmíněná	Get_Attribute_All	vrátí předdefinovaný výpis atributů objektu
10hex	n/a*	podmíněná	Set_Attribute_Single	nastaví hodnotu atributu
11hex	volitelná	n/a	Find_Next_Object_Instance	vrátí seznam ID stávajících instancí identifikačního objektu
18hex	volitelná	podmíněná	Get_Member	vrátí vybranou část atributu (u atributů typu string s národními znaky(STRINGI))

* n/a – nelze vztáhnout na daný prvek (*angl. not applicable*)

Popis jednotlivých služeb objektu, jejich žádostí a odpovědí

Pro možnost popisu výše jmenovaných služeb objektu WSDL dokumentem je nutné znát, jaká data (typ žádosti) daná služba přijímá a také jaká data (typ odpovědi) služba odesílá. V následujícím přehledu jsou tedy stručné popisy služeb Identifikačního objektu, jejich žádostí a odpovědí.

3.4.2.1 Služba Get_Attribute_Single

Služba vrací hodnotu jednoho konkrétního atributu specifikovaného v žádosti. Pokud je detekovaná chyba, služba vrací chybové hlášení.

Žádost:

Tabulka 6: Data přijímaná službou Get_Attribute_Single jako žádost

jméno	CIP datový typ	popis parametru
ID číslo atributu	USINT	označuje ID atributu, jehož hodnota se má vrátit

Odpověď:

Tabulka 7: Data odesílaná službou Get_Attribute_Single jako úspěšná odpověď

jméno	CIP datový typ	popis parametru
Data atributu	typ požadovaného atributu	obsahuje data požadovaného atributu

3.4.2.2 Služba Reset

Služba vyvolá reset zařízení či jeho části (to v případě volání služby na úrovni instance, kde instance představuje určitou část daného zařízení), kterou Identifikační objekt reprezentuje. Ve volání služby je definovaný typ resetu, který se má nad zařízením (příp. jeho částí) provést. Pokud je detekovaná chyba, služba vrací chybové hlášení.

Pokud Identifikační objekt obdrží požadavek na reset zařízení:

- rozhodne zda může poskytnout daný typ resetu (specifický pro požadovanou akci)
- odpoví na žádost
- pokusí se vykonat požadovaný typ resetu

Žádost:

Tabulka 8: Data přijímaná službou Reset jako žádost

jméno	CIP datový typ	popis parametru
typ	USINT	typ požadovaného resetu*

***** různé typy resetu daného objektu jsou definovány ve specifikaci protokolu [2]

Odpověď:

Specifikace protokolu CIP nedefinuje žádná konkrétní data v odpovědi služby *Reset* pro Identifikační objekt. Standardní formát odpovědi obsahuje tedy pouze stavový kód potvrzující úspěšné provedení resetu.

3.4.2.3 Služba Get_Attribute_All

Služba vrací všechny hodnoty atributů třídy nebo instance definované v popisu daného objektu. Jak je popsáno dále v kapitole 4, rozlišení služby pracující na úrovni třídy či instance je klíčové pro její popis pomocí WSDL dokumentu, kde musí být jednoznačně rozlišeno na

jaké úrovni se daná služba volá. Odpověď na volání služby na úrovni třídy vrací výše jmenované atributy třídy (viz.*Tabulka 3*) a odpověď na volání na úrovní instance vrací atributy instance (viz.*Tabulka 4*). Je zřejmé, že tyto dvě odpovědi nebudou mít stejnou délku (různý počet parametrů). Toto musí být opět specifikováno ve WSDL popisu.

Žádost:

Specifikace protokolu CIP nedefinuje žádná data obsažená v žádosti o službu *Get_Attribute_All.* Standardní formát žádosti obsahuje tedy jen kód dané služby.

Odpověď:

Tabulka 9: Data odesílaná službou Get_Attribute_All jako úspěšná odpověď

jméno	CIP datový typ	popis parametru
data atributů	struktura obsahující data všech atributů	data všech atributů specifikovaných danou třídou nebo instancí

3.4.2.4 Služba Set_Attribute_Single

Služba změní hodnotu specifikovaného atributu. Pokud nastane chyba, služba vrací chybové hlášení. Tato služba je definována pouze pro úroveň instance.

Žádost:

Tabulka 10: Data přijímaná službou Set_Attribute_Single jako žádost

jméno	CIP datový typ	popis parametru
ID číslo atributu	USINT	označuje atribut který má být změněn
Data atributu	dle požadovaného atributu	obsahuje hodnotu na kterou má být požadovaný atribut změněn

Odpověď:

Specifikace protokolu CIP nedefinuje žádná konkrétní data v odpovědi služby *Set_Attribute_Single* pro Identifikační objekt. Standardní formát odpovědi obsahuje tedy pouze stavový kód, potvrzující úspěšnou změnu hodnoty atributu.

3.4.2.5 Služba Find_Next_Object_Instance

Tato služba je definována pouze na úrovni třídy. Služba vrátí seznam identifikačních čísel vytvořených instancí. Pokud je detekována chyba, služba vrací příslušné chybové hlášení. Jinak služba vrací úspěšnou odpověď.

Žádost:

Tabulka 11: Data přijímaná službou Find_Next_Object_Instance jako žádost

jméno	CIP datový typ	popis parametru
Maximální vrácená hodnota	USINT	označuje maximální počet ID instancí k vrácení v odpovědi na žádost

Odpověď:

Tabulka 12: Data odesílaná službou Find_Next_Object_Instance jako úspěšná odpověď

jméno	CIP datový typ	popis parametru
Počet vrácených ID instancí	USINT	obsahuje počet vrácených ID instancí specifikovaných v dané odpovědi
Seznam ID instancí	pole UINT	obsahuje seznam ID instancí. ID jsou vráceny v poli 16 bit přirozených čísel

3.4.2.6 Služba Get_Member

Jednotlivé atributy se mohou skládat z polí základních datových typů. Služba *Get_Member* umožňuje přečtení jednotlivé položky pole specifikovaného atributu. Pokud je detekovaná chyba (např. požadavek na čtení položky mimo rozsah pole), služba vrací chybové hlášení.

Žádost:

Tabulka 13: Data přijímaná službou Get_Member jako žádost

jméno	CIP datový typ	popis parametru
ID číslo atributu	USINT	označuje atribut z kterého má být přečtena položka pole
ID položky	UINT	označuje položku z atributu, která má být přečtena

Odpověď:

Tabulka 14: Data odesílaná službou Get_Member jako úspěšná odpověď

jméno	CIP datový typ	popis parametru
ID položky	UINT	označuje přečtenou položku z atributu
Data položky	dle požadovaného položky	obsahuje data požadované položky

4 Společný popis služeb řídicího a informačního systému

V této kapitole jsou navrženy varianty mapování výše popsaných modelů služeb, tedy modelů definovaných standardy WSDL 2.0 a CIP. Mapování spočívá obecně v nalezení analogií mezi jednotlivými částmi těchto modelů. V tomto případě jde o porovnání dvou navržených modelů služeb a hledaní vzájemně si odpovídajících komponent těchto modelů. S pomocí tohoto srovnání je pak již možné popsat jeden systém pomocí systému druhého a samozřejmě v ideálním případě i naopak. Zde je třeba podotknout, že druhá varianta, tedy možnost popsat službu WSDL pomocí CIP služby, není cílem této práce. V tomto konkrétním případě jde o navržení popisu služeb CIP pomocí WSDL 2.0.

4.1 Mapování modelů služeb

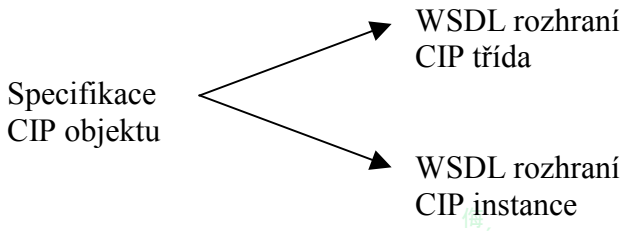
V předchozích kapitolách byly popsány modely služeb podnikového a průmyslového systému. V této části jsou navrženy dvě varianty možného mapování příslušných částí daných modelů (viz. *Obrázek 13*). Mapování je definováno na abstraktní úrovni těchto modelů. Pro konkrétní realizaci je nutné tento proces rozšířit i o konkrétní vazby (*angl. Bindings*), tak aby pak bylo možné konkrétní službu zavolat přes požadované IT rozhraní podnikového systému.

První varianta mapování vychází z toho, že model služeb standardu CIP je rozdělen do dvou WSDL rozhraní (*angl. Interface*). První rozhraní v tomto případě reprezentuje CIP třídu (*angl. Class*) a její příslušné služby, které jsou definované právě na úrovni třídy (*angl. Class level* viz. výše). Druhé rozhraní reprezentuje CIP instanci (*angl. Instance*) opět s danými službami, které pracují v tomto případě na úrovni instance (*angl. Instance level*). Mapování dalších částí modelů pak již probíhá v poměru jedna k jedné, jak je znázorněno na *Obrázku 14*. CIP služba (*angl. Service*) je v tomto modelu reprezentována WSDL operací

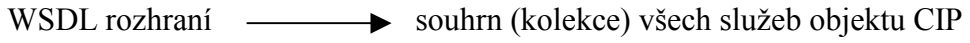
(*angl. Operation*), žádost CIP (*angl. Request Message*) odpovídá vstupní zprávě WSDL (*angl. Input*), odpověď CIP (*angl. Response Message*) zastupuje výstupní zprávu WSDL (*angl. Output*). Chybová hlášení CIP (*angl. Error Response*) jsou mapována na chybové zprávy WSDL (*angl. Fault*). Datové typy které používá protokol CIP (*angl. CIP Data Types*) je také nutné mapovat na příslušné datové typy WSDL (*angl. Types*). Mapování datových typů se věnuje detailněji kapitola 4.2.

Navržená varianta mapování je přehledná, srozumitelná a její konkrétní realizace je snadná. Jejím hlavním nedostatkem je ovšem fakt, že pro každý CIP objekt je nutné vytvořit dvě samostatné WSDL rozhraní, které spolu nejsou nijak svázány. Tímto se popis objektu jako celku rozděluje vlastně do dvou samostatných částí. Z tohoto důvodu byla navržena druhá varianta mapování.

Varianta I:

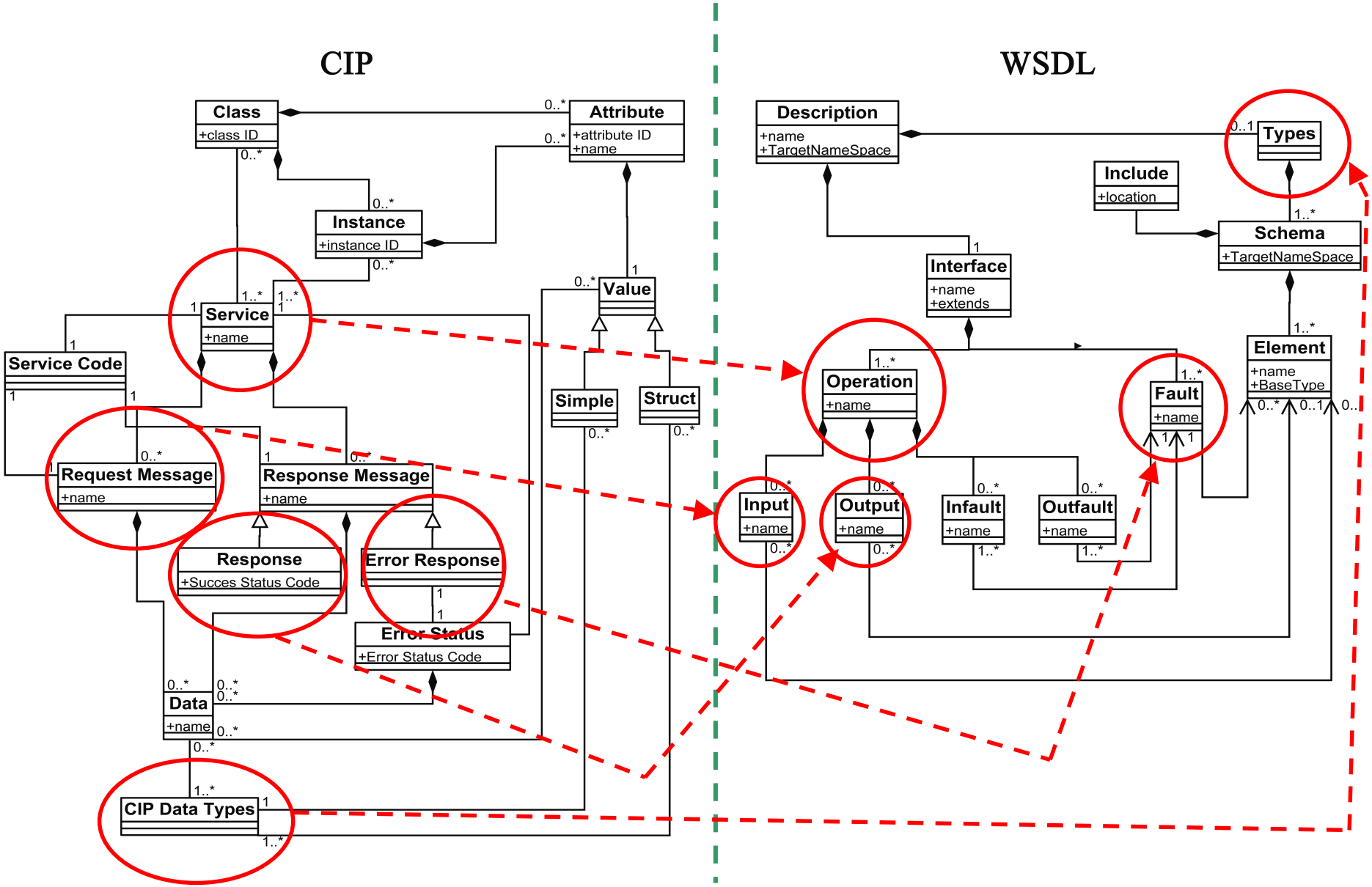


Varianta II:



Obrázek 13: Varianty mapování

Druhá varianta řeší nedostatek předchozího návrhu tím, že celý CIP objekt popisuje jedním WSDL rozhraním. Rozhraní tohoto WSDL dokumentu zahrnuje soubor veškerých služeb daného objektu. Každá služba musí mít v tomto případě přímo definovanou svou příslušnost buď k třídě nebo k instanci. Toto je možné řešit například předponou nebo příponou v názvu WSDL operace. Tím se jasně definuje, na které úrovni má být daná CIP služba volána. Služba pracující na úrovni třídy běžně vrací jiná data než služba, která pracuje na úrovni instance, jak bude ukázáno v následujících kapitolách popisujících konkrétní příklady společného popisu. Pro tuto variantu byla zvoleno použití předpon k názvům operace.



Obrázek 14: Mapování modelů služeb

Použití prefixu lze totiž pak snadno využít například třídění operací a tak následně jednoduše pracovat jen s operacemi, které přísluší buď třídě nebo instanci. Mapování dalších částí standardu probíhá i v tomto případě stejně jako v první variantě.

Ze srovnání obou navržených variant mapovaní je zřejmé, že každá varianta má své výhody a nevýhody. V první variantě je systém, který bude daný objekt a jeho služby využívat, nucen pracovat s dvěma WSDL rozhraními a tím může docházet k jistým komplikacím během implementace. V druhém případě mapování je dodržena logická kombinace – jeden CIP objekt = jedno WSDL rozhraní a tím možnost snadnějšího použití takové metody. Vyskytuje se zde však problém zdvojení definic operací lišících se pouze předponou, tím že každý WSDL dokument musí v tomto případě obsahovat dvě definice té samé služby i pro případy, kdy je její chování pro obě úrovně identické. Tato varianta naopak obsahuje výše popsanou výhodu snadného třídění operací tříd a instancí.

Z výše popsaného plyne, že obě varianty jsou víceméně rovnocenné a jejich možné použití závisí na konkrétních možnostech implementace daného problému.

4.2 Mapování datových typů

V této kapitole je uvedeno, jakým způsobem lze mapovat datové typy používané protokolem CIP na datové typy s kterými pracuje jazyk WSDL. Jelikož jazyk WSDL vychází z formátu XML, používá stejně jako jazyk XML schémata pro svůj popis. XML schéma popisuje mimo jiné právě datové typy s kterými daný dokument může pracovat. První možností je tedy mapovat CIP datové typy (*angl. CIP Data Types*) na příslušné typy definované daným standardním XML schématem. Druhou variantou je vytvoření nového XML schématu (např. CIP Namespace) a ten pak použít v rámci WSDL dokumentu.

Nalezení analogií mezi CIP datovými typy a standardním XML schématem [30] popisuje následující tabulka (viz. *Tabulka 15*) do níž byly z důvodu přehledností vybrány pouze základní a nejčastěji používané datové typy.

Tabulka 15: Mapování datových typů

CIP Data Types				XML schema			
		Rozsah				rozsah	
typ	**popis**	**min**	**max**	**typ**	**popis**	**min**	**max**
BOOL	logická hodnota	true, false, 1, 0		boolean	logická hodnota	true, false, 1, 0	
SINT	celé číslo 8bit	-128	127	byte	celé číslo 8bit	-128	127
INT	celé číslo 16bit	-32768	32767	short	celé číslo 16bit	-32768	32767
DINT	celé číslo 32bit	-2^{31}	$2^{31}-1$	int	celé číslo 32bit	-2^{31}	$2^{31}-1$
LINT	celé číslo 64bit	-2^{63}	$2^{63}-2$	long	celé číslo 64bit	-2^{63}	$2^{63}-2$
USINT	nezáporné celé číslo 8bit	0	255	unsignedByte	nezáporné celé číslo 8bit	0	255
UINT	nezáporné celé číslo 16bit	0	65535	unsignedShort	nezáporné celé číslo 16bit	0	65535
UDINT	nezáporné celé číslo 32bit	0	$2^{32}-1$	unsignedInt	nezáporné celé číslo 32bit	0	$2^{32}-1$
ULINT	nezáporné celé číslo 64bit	0	$2^{64}-1$	unsignedLong	nezáporné celé číslo 64bit	0	$2^{64}-1$
REAL	desetinné číslo	dle IEEE 754		float	desetinné číslo	32bitové číslo v plovoucí desetinné čárce	
TIME	délka časového intervalu	DINT v ms		duration	délka časového intervalu		
STRING	řetězec znaků			string	řetězec znaků		

5 Příklady společného popisu

V této kapitole bude na příkladu objektu specifikace CIP a jeho službách ukázáno, jak je možné takový objekt popsat pomocí WSDL dokumentu. S pomocí tohoto popisu bude služba CIP objektu přístupná z libovolného IT prostředí a bude jí možné zavolat jako běžnou síťovou službu. Návrhy možných způsobů komunikace a přístupu k jednotlivým službám z IT prostředí řeší samostatná kapitola 5.2.

5.1 Konkrétní realizace WSDL dokumentu

V této kapitole bude ukázáno jak lze sestavit konkrétní WSDL dokument popisující CIP služby daného objektu. Skladba a jednotlivé části WSDL dokumentu jsou ve všech případech shodné s příkladem uvedeným v kapitole 2 této práce. Popis bude prezentován ve dvou variantách, jak bylo navrženo v kapitole 4 této práce a to na příkladu Identifikačního objektu popsaného v kapitole 3.4.

5.1.1 WSDL popis služeb Identifikačního objektu – Varianta I

Jak bylo navrženo, v této variantě je WSDL popis rozdělen na dva dokumenty popisující služby pracující na úrovni třídy a služby pracující na úrovni instance.

WDSL popis atributů třídy a služeb pracujících na úrovni třídy objektu:

Obálka WSDL dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
xmlns:wsdl=http://www.w3.org/ns/wsdl
targetNamespace="urn:wsdlSamples"
xmlns:tns="urn:wsdlSamples"
xmlns:dtns="urn:wsdlSamplesData">
```

Definice atributů třídy objektu, datových formátů žádostí a odpovědí služeb:

Atributy a jejich datové typy odpovídají specifikaci Identifikačního objektu, tak jak je prezentována v kapitole 3. Definice formátu žádostí a odpovědí jednotlivých služeb opět odpovídá specifikacím z kapitoly 3.

```
<wsdl:types>
<xs:schema
xmlns:xs=http://www.w3.org/2001/XMLSchema
targetNamespace="urn:wsdlSamplesData">
```

Atributy třídy:

```
<xs:complexType name="MaxInstace">
  <xs:sequence>
    <xs:element name="MaxInstance" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Revision">
  <xs:sequence>
    <xs:element name="Revision" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Number_of_Instaces">
  <xs:sequence>
    <xs:element name="Number_of_Instaces" type="xs: unsignedShort "/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MaxID_Number_Class_Attributes">
  <xs:sequence>
    <xs:element name="MaxID_Number_Class_Attributes"
       type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MaxID_Number_Instance_Attributes">
  <xs:sequence>
    <xs:element name="MaxID_Number_Instance_Attributes"
       type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="Optional_Attribute_List">
  <xs:sequence>
    <xs:element name="Number_of_Attributes">
       type="xs:unsignedShort"/>
    <xs:element name="Optional_Attributes" type="xs:unsignedShort"
       minOccurs="0"
       maxOccurs="Max_Returned_Values"
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Optional_Service_List">
  <xs:sequence>
    <xs:element name="Number_of_Services">
       type="xs:unsignedShort"/>
    <xs:element name="Optional_Services" type="xs:unsignedShort"
       minOccurs="0"
       maxOccurs="Max_Returned_Values"
  </xs:sequence>
</xs:complexType>
```

Datové formáty žádostí a odpovědí služeb:

Jak bylo popsáno v kapitole 3.4.2 operace *Get_Attribute_All* nevyžaduje žádná vstupní data, takže její zpráva žádosti je reprezentována prázdnou zprávou bez specifických dat pouze s kódem dané CIP služby (*angl. Service Code*). Obdobně je tomu u odpovědí u operací *Reset* a *Set_Attribute_Single,* jejichž odpovědi jsou prázdné a služba odesílá, pokud nenastala chyba, pouze potvrzení úspěšného vykonání dané služby (*angl. Success Status Code*). Kód služby a potvrzení úspěšného vykonání jsou samozřejmě součástí všech ostatních žádostí/odpovědí, které mimo tyto obsahují konkrétní definovaná data.

```
<xs:complexType name="Get_Atribute_Single_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="AttributeID" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Reset_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Reset_Type" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Reset_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code"type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>

<xs:complexType name="Get_Atribute_All_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>
```

```
    <xs:complexType name="Get_Atribute_All_Response">
      <xs:sequence>
        <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
        <xs:element ref="MaxInstace"/>
        <xs:element ref="Revision"/>
        <xs:element ref="Number_of_Instaces"/>
        <xs:element ref="Optional_Attribute_List"/>
        <xs:element ref="Optional_Service_List"/>
        <xs:element ref="MaxID_Number_Class_Attributes"/>
        <xs:element ref="MaxID_Number_Instance_Attributes"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Find_Next_Object_Instance_Request">
      <xs:sequence>
        <xs:element name="Service_Code" type="xs:unsignedByte"/>
        <xs:element name="Max_Returned_Values" type="xs:unsignedByte"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Find_Next_Object_Instance_Response">
      <xs:sequence>
        <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
        <xs:element name="Number_of_List_Members"
           type="xs:unsignedByte"/>
        <xs:element name="Instance_ID_List" type="xs:unsignedShort"
           minOccurs="0" maxOccurs="Max_Returned_Values"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Get_Member_Request">
      <xs:sequence>
        <xs:element name="Service_Code" type="xs:unsignedByte"/>
        <xs:element name="Attribute_ID" type="xs:unsignedShort"/>
        <xs:element name="Member_ID" type="xs:short"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Get_Member_Response">
      <xs:sequence>
        <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
        <xs:element name="Memeber_ID" type="xs:unsignedShort"/>
        <xs:element name="Memeber_Data" type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ObjectClassError">
      <xs:sequence>
        <xs:element name="Error" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>

    </xs:schema>
  </wsdl:types>
```

Rozhraní třídy objektu:

```
<wsdl:interface name="IdentityObjectClassInterface">
<wsdl:fault name = "Error" element = "dtns:ObjectClassError"/>
```

Definice služeb třídy objektu:

Operace standardu WSDL mohou být několika typů: vstupně-výstupní (*angl. in-out*), pouze vstup (*angl. in-only*), pouze výstup (*angl. out-only*). V tomto případě, kdy služba standardu CIP v každém případě vrací a přijímá nějaká data, je použita pouze operace typu vstupně-výstupní (*angl. in-out*). Pro chybová hlášení je použit obecný řetězec (*angl. string*), do kterého je vložena příslušná chybová zpráva protokolu CIP.

```
<wsdl:operation name="Get_Atrribute_Single"
  pattern="http://www.w3.org/ns/wsdl/in-out">
    <wsdl:input messageLabel="In"
      element="dtns:Get_Atribute_Single_Request"/>
    <wsdl:output messageLabel="Out" element="dtns:MaxInstace"/>
    <wsdl:output messageLabel="Out" element="dtns:Revision"/>
    <wsdl:output messageLabel="Out" element="dtns:Number_of_Instaces"/>
    <wsdl:output messageLabel="Out"
      element="dtns:Optional_Attribute_List"/>
    <wsdl:output messageLabel="Out"
       element="dtns:Optional_Service_List "/>
    <wsdl:output messageLabel="Out"
      element="dtns:MaxID_Numeber_Class_Attributes "/>
    <wsdl:output messageLabel="Out"
      element="dtns:MaxID_Numeber_Instance_Attributes"/>
    <wsdl:outfault messageLabel="Out" ref="dtns:ObjectClassError"/>
</wsdl:operation>

<wsdl:operation name="Reset"
  pattern="http://www.w3.org/ns/wsdl/in-out">
  <wsdl:input messageLabel="In" element="dtns:Reset_Request"/>
  <wsdl:output messageLabel="Out" element="dtns:Reset_Response"/>
  <wsdl:outfault messageLabel="Out" ref="dtns:ObjectClassError"/>
</wsdl:operation>

<wsdl:operation name="Get_Atrribute_All"
  pattern="http://www.w3.org/ns/wsdl/in-out ">
  <wsdl:input messageLabel="In" element="dtns:Get_Atribute_All_Request"/>
  <wsdl:output messageLabel="Out"
    element="dtns:Get_Atribute_All_Response"/>
  <wsdl:outfault messageLabel="Out" ref="dtns:ObjectClassError"/>
</wsdl:operation>

<wsdl:operation name="Find_Next_Object_Instance"
  pattern="http://www.w3.org/ns/wsdl/in-out">
  <wsdl:input messageLabel="In"
    element="dtns:Find_Next_Object_Instance_Request"/>
  <wsdl:output messageLabel="Out"
    element="dtns:Find_Next_Object_Instance_Response"/>
  <wsdl:outfault messageLabel="Out" ref="dtns:ObjectClassError"/>
</wsdl:operation>
```

```
  <wsdl:operation name="Get_Member"
    pattern="http://www.w3.org/ns/wsdl/in-out">
    <wsdl:input messageLabel="In" element="dtns:Get_Member_Request"/>
    <wsdl:output messageLabel="Out" element="dtns:Get_Member_Response"/>
    <wsdl:outfault messageLabel="Out" ref="dtns:ObjectClassError"/>
  </wsdl:operation

</wsdl:interface>
```

Definice vazeb třídy objektu (TCP a CIP):

Pro možnost použití dané služby, jak z průmyslové, tak z IT domény, je nutné do popisu zahrnout, jak komunikační vazbu na průmyslový protokol CIP, tak i na protokol TCP. Možné způsoby komunikace jsou popsány v kapitole 5.2.

```
  <wsdl:binding name="TCPBinding"
    interface="tns: IdentityObjectClassInterface">
    <wsdl:operation ref="tns:Get_Atrribute_Single"/>
    <wsdl:operation ref="tns:Reset"/>
    <wsdl:operation ref="tns:Get_Atrribute_All"/>
    <wsdl:operation ref="tns:Find_Next_Object_Instance"/>
    <wsdl:operation ref="tns:Get_Member"/>
  </wsdl:binding>

  <wsdl:binding name="CIPBinding"
    interface="tns: IdentityObjectClassInterface">
    <wsdl:operation ref="tns:Get_Atrribute_Single"/>
    <wsdl:operation ref="tns:Reset"/>
    <wsdl:operation ref="tns:Get_Atrribute_All"/>
    <wsdl:operation ref="tns:Find_Next_Object_Instance"/>
    <wsdl:operation ref="tns:Get_Member"/>
  </wsdl:binding>

  <wsdl:service name="IdentityObjectClass"
    interface="tns: IdentityObjectClassInterface">
    <wsdl:endpoint name="IdentityObjectClassEndPoint"
    binding="tns:TCPBinding"
    address="tcp://localhost/IdentityObjectClass"/>
    <wsdl:endpoint name="IdentityObjectClassEndPoint"
    binding="tns:CIPBinding"
    address="cip://localhost/IdentityObjectClass"/>
  </wsdl:service>

</wsdl:description>
```

WDSL popis atributů instance a služeb pracujících na úrovni instance objektu:

Obálka WSDL dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
xmlns:wsdl="http://www.w3.org/ns/wsdl"
targetNamespace="urn:wsdlSamples"
xmlns:tns="urn:wsdlSamples"
xmlns:dtns="urn:wsdlSamplesData">
```

Definice povinných atributů instance objektu, datových formátů žádostí a odpovědí služeb:

```
<wsdl:types>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
targetNamespace="urn:wsdlSamplesData">

   <xs:complexType name="VendorID">
     <xs:sequence>
       <xs:element name="VendorID" type="xs:unsignedShort "/>
     </xs:sequence>
   </xs:complexType>

   <xs:complexType name="DeviceType">
     <xs:sequence>
       <xs:element name="DeviceType" type="xs:unsignedShort "/>
     </xs:sequence>
   </xs:complexType>

   <xs:complexType name="ProductCode">
     <xs:sequence>
       <xs:element name="ProductCode" type="xs:unsignedShort"/>
     </xs:sequence>
   </xs:complexType>

   <xs:complexType name="Revision">
     <xs:sequence>
       <xs:element name="MajorRevision" type="xs:unsignedByte"/>
       <xs:element name="MinorRevision" type="xs:unsignedByte"
     </xs:sequence>
   </xs:complexType>

   <xs:complexType name="Status">
     <xs:sequence>
       <xs:element name="Status" type="xs:int"/>
     </xs:sequence>
   </xs:complexType>

   <xs:complexType name="SerialNumber">
     <xs:sequence>
       <xs:element name="SerialNumber" type="xs:unsignedInt"/>
     </xs:sequence>
   </xs:complexType>

   <xs:complexType name="ProductName">
     <xs:sequence>
       <xs:element name="ProductName" type="xs:stirng"/>
     </xs:sequence>
   </xs:complexType>
```

Datové formáty žádostí a odpovědí služeb:

```
   <xs:complexType name="Get_Atribute_Single_Request">
     <xs:sequence>
       <xs:element name="Service_Code" type="xs:unsignedByte"/>
       <xs:element name="AttributeID" type="xs:unsignedByte"/>
     </xs:sequence>
   </xs:complexType>
```

```
<xs:complexType name="Reset_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Reset_Type" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Reset_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>

<xs:complexType name="Get_Atribute_All_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>

<xs:complexType name="Get_Atribute_All_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
    <xs:element ref="VendorID"/>
    <xs:element ref="DeviceType"/>
    <xs:element ref="ProductCode"/>
    <xs:element ref="Revision"/>
    <xs:element ref="Status"/>
    <xs:element ref="SerialNumber"/>
    <xs:element ref="ProductName"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Set_Attribute_Single_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Attribute_ID" type="xs:unsignedShort"/>
    <xs:element name="Attribute_Data" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Set_Attribute_Single_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>

<xs:complexType name="Get_Member_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Attribute_ID" type="xs:unsignedShort"/>
    <xs:element name="Member_ID" type="xs:short"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Get_Member_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
    <xs:element name="Memeber_ID" type="xs:unsignedShort"/>
    <xs:element name="Memeber_Data" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>
```

```
   <xs:complexType name="ObjectInstanceError">
     <xs:sequence>
       <xs:element name="Error" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>

   </xs:schema>
</wsdl:types>
```

Rozhraní instance objektu:

```
<wsdl:interface name="IdentityObjectInstanceInterface">
<wsdl:fault name = "Error" element = "dtns:ObjectInstanceError"/>
```

Definice služeb instance objektu:

```
   <wsdl:operation name="Get_Atrribute_Single"
     pattern="http://www.w3.org/ns/wsdl/in-out">
       <wsdl:input messageLabel="In"
          element="dtns:Get_Atribute_Single_Request"/>
       <wsdl:output messageLabel="Out" element="dtns:VendorID"/>
       <wsdl:output messageLabel="Out" element="dtns:DeviceType"/>
       <wsdl:output messageLabel="Out" element="dtns:ProductCode"/>
       <wsdl:output messageLabel="Out" element="dtns:Revision"/>
       <wsdl:output messageLabel="Out" element="dtns:Status"/>
       <wsdl:output messageLabel="Out" element="dtns:SerialNumber"/>
       <wsdl:output messageLabel="Out"element="dtns:ProductName"/>
       <wsdl:outfault messageLabel="Out" ref="dtns:ObjectInstanceError"/>
   </wsdl:operation>

   <wsdl:operation name="Reset"
     pattern="http://www.w3.org/ns/wsdl/in-out">
     <wsdl:input messageLabel="In" element="dtns:Reset_Request"/>
     <wsdl:output messageLabel="Out" element="dtns:Reset_Response"/>
     <wsdl:outfault messageLabel="Out" ref="dtns:ObjectInstanceError"/>
   </wsdl:operation>

   <wsdl:operation name="Get_Atrribute_All"
     pattern="http://www.w3.org/ns/wsdl/in-out">
       <wsdl:input messageLabel="In"
          element="dtns:Get_Atribute_All_Request"/>
       <wsdl:output messageLabel="Out"
          element="dtns:Get_Atribute_All_Response"/>
       <wsdl:outfault messageLabel="Out" ref="dtns:ObjectInstanceError"/>
   </wsdl:operation>

   <wsdl:operation name="Set_Attribute_Single"
     pattern="http://www.w3.org/ns/wsdl/in-out">
       <wsdl:input messageLabel="In"
          element="dtns:Set_Attribute_Single_Request"/>
      <wsdl:output messageLabel="Out"
          element="dtns:Set_Attribute_Single_Response"/>
       <wsdl:outfault messageLabel="Out" ref="dtns:ObjectInstanceError"/>
   </wsdl:operation>

   <wsdl:operation name="Get_Member"
     pattern="http://www.w3.org/ns/wsdl/in-out">
       <wsdl:input messageLabel="In" element="dtns:Get_Member_Request"/>
       <wsdl:output messageLabel="Out" element="dtns:Get_Member_Response"/>
       <wsdl:outfault messageLabel="Out" ref="dtns:ObjectInstanceError"/>
   </wsdl:operation>
</wsdl:interface>
```

Definice vazeb instance objektu (TCP a CIP):

```
<wsdl:binding name="TCPBinding"
  interface="tns: IdentityObjectInstanceInterface">
    <wsdl:operation ref="tns:Get_Atrribute_Single"/>
    <wsdl:operation ref="tns:Reset"/>
    <wsdl:operation ref="tns:Get_Atrribute_All"/>¨
    <wsdl:operation ref="tns:Set_Attribute_Single"/>
    <wsdl:operation ref="tns:Get_Member"/>
</wsdl:binding>

<wsdl:binding name="CIPBinding"
  interface="tns: IdentityObjectInstanceInterface">
    <wsdl:operation ref="tns:Get_Atrribute_Single"/>
    <wsdl:operation ref="tns:Reset"/>
    <wsdl:operation ref="tns:Get_Atrribute_All"/>
    <wsdl:operation ref="tns:Set_Attribute_Single"/>
    <wsdl:operation ref="tns:Get_Member"/>
</wsdl:binding>

<wsdl:service name="IdentityObjectInstance"
  interface="tns: IdentityObjectInstanceInterface">
  <wsdl:endpoint name="IdentityObjectInstanceEndPoint"
  binding="tns:TCPBinding"
  address="tcp://localhost/IdentityObjectInstance"/>
  <wsdl:endpoint name="IdentityObjectInstanceEndPoint"
  binding="tns:CIPBinding"
  address="cip://localhost/IdentityObjectInstance"/>
</wsdl:service>

</wsdl:description>
```

5.1.2 WSDL popis služeb Identifikačního objektu – Varianta II

Jak bylo navrženo v kapitole 4, v této variantě je WSDL dokument společný pro služby třídy i služby instance. U každé služby je pak nutné definovat, zda má být volána na úrovní třídy nebo na úrovní instance.

Obálka WSDL dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
xmlns:wsdl=http://www.w3.org/ns/wsdl
targetNamespace="urn:wsdlSamples"
xmlns:tns="urn:wsdlSamples"
xmlns:dtns="urn:wsdlSamplesData">
```

Definice povinných atributů Identifikačního objektu, datových formátů žádostí a odpovědí služeb:

Také pro WSDL popis celého objektu je nutné definovat všechny atributy, které daný objekt využívá. V tomto případě jak na úrovni třídy, tak na úrovni instance. Také je nutné definovat datové formáty odpovědí a žádostí služeb pro obě úrovně.

```
<wsdl:types>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
targetNamespace="urn:wsdlSamplesData">
```

Atributy třídy:

```
<xs:complexType name="MaxInstace">
  <xs:sequence>
    <xs:element name="MaxInstance" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Revision">
  <xs:sequence>
    <xs:element name="Revision" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Number_of_Instaces">
  <xs:sequence>
    <xs:element name="Number_of_Instaces" type="xs: unsignedShort "/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Optional_Attribute_List">
  <xs:sequence>
    <xs:element name="Number_of_Attributes">
       type="xs:unsignedShort"/>
   <xs:element name="Optional_Attributes" type="xs:unsignedShort"
       minOccurs="0" maxOccurs="Max_Returned_Values"
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Optional_Service_List">
  <xs:sequence>
    <xs:element name="Number_of_Services"> type="xs:unsignedShort"/>
    <xs:element name="Optional_Services" type="xs:unsignedShort"
       minOccurs="0" maxOccurs="Max_Returned_Values"
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MaxID_Number_Class_Attributes">
  <xs:sequence>
    <xs:element name="MaxID_Number_Class_Attributes"
       type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MaxID_Number_Instance_Attributes">
  <xs:sequence>
    <xs:element name="MaxID_Number_Instance_Attributes"
       type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>
```

Atributy instance:

```
<xs:complexType name="VendorID">
  <xs:sequence>
    <xs:element name="VendorID" type="xs:unsignedShort "/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="DeviceType">
  <xs:sequence>
    <xs:element name="DeviceType" type="xs:unsignedShort "/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ProductCode">
  <xs:sequence>
    <xs:element name="ProductCode" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Revision">
  <xs:sequence>
    <xs:element name="MajorRevision" type="xs:unsignedByte"/>
    <xs:element name="MinorRevision" type="xs:unsignedByte"
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Status">
  <xs:sequence>
    <xs:element name="Status" type="xs:int"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SerialNumber">
  <xs:sequence>
    <xs:element name="SerialNumber" type="xs:unsignedInt"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ProductName">
  <xs:sequence>
    <xs:element name="ProductName" type="xs:stirng"/>
  </xs:sequence>
</xs:complexType>
```

Datové formáty žádostí a odpovědí služeb:

```
<xs:complexType name="Get_Atribute_Single_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="AttributeID" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Reset_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Reset_Type" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Reset_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>
```

```
<xs:complexType name="Get_Atribute_All_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>

<xs:complexType name="Class_Get_Atribute_All_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
    <xs:element ref="MaxInstace"/>
    <xs:element ref="Revision"/>
    <xs:element ref="Number_of_Instaces"/>
    <xs:element ref="Optional_Attribute_List"/>
    <xs:element ref="Optional_Service_List "/>
    <xs:element ref="MaxID_Number_Class_Attributes "/>
    <xs:element ref="MaxID_Number_Instance_Attributes"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Instance_Get_Atribute_All_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
    <xs:element ref="VendorID"/>
    <xs:element ref="DeviceType"/>
    <xs:element ref="ProductCode"/>
    <xs:element ref="Revision"/>
    <xs:element ref="Status"/>
    <xs:element ref="SerialNumber"/>
    <xs:element ref="ProductName"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Set_Attribute_Single_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Attribute_ID" type="xs:unsignedShort"/>
    <xs:element name="Attribute_Data" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Set_Attribute_Single_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
  <xs:sequence/>
</xs:complexType>

<xs:complexType name="Find_Next_Object_Instance_Request">
  <xs:sequence>
    <xs:element name="Service_Code" type="xs:unsignedByte"/>
    <xs:element name="Max_Returned_Values" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Find_Next_Object_Instance_Response">
  <xs:sequence>
    <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
    <xs:element name="Number_of_List_Members" type="xs:unsignedByte"/>
    <xs:element name="Instance_ID_List" type="xs:unsignedShort"
       minOccurs="0" maxOccurs="Max_Returned_Values"/>
  </xs:sequence>
</xs:complexType>
```

```
  <xs:complexType name="Get_Member_Request">
    <xs:sequence>
      <xs:element name="Service_Code" type="xs:unsignedByte"/>
      <xs:element name="Attribute_ID" type="xs:unsignedShort"/>
      <xs:element name="Member_ID" type="xs:short"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Get_Member_Response">
    <xs:sequence>
      <xs:element name="Success_Status_Code" type="xs:unsignedByte"/>
      <xs:element name="Memeber_ID" type="xs:unsignedShort"/>
      <xs:element name="Memeber_Data" type="xs:anyType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ObjectIdentityError">
    <xs:sequence>
      <xs:element name="Error" type="xs:string"/>
    </xs:sequence>
  </xs:complexType

  </xs:schema>
</wsdl:types>
```

Rozhraní objektu:

```
<wsdl:interface name="IdentityObjectInterface">
<wsdl:fault name = "Error" element = "dtns:ObjectIdentityError"/>
```

Definice služeb objektu:

Definice služeb je zdvojena v případech, kdy je daná služba požadována na obou úrovních objektu. Služba pak obsahuje označení, které specifikuje na jaké úrovni má být volána. Toto je řešeno pomocí předpon v názvu dané služby a sice předponou *„Class_“* pro třídu a předponou *„Instance_“* pro instanci. Odpovědi služeb se v těchto případech liší, což musí být opět specifikováno ve WSDL popisu.

```
  <wsdl:operation name="Class_Get_Atrribute_Single"
    pattern="http://www.w3.org/ns/wsdl/in-out">
      <wsdl:input messageLabel="In"
         element="dtns:Get_Atribute_Single_Request"/>
      <wsdl:output messageLabel="Out" element="dtns:MaxInstace"/>
      <wsdl:output messageLabel="Out" element="dtns:Revision"/>
      <wsdl:output messageLabel="Out" element="dtns:Number_of_Instaces"/>
      <wsdl:output messageLabel="Out"
         element="dtns:Optional_Attribute_list"/>
      <wsdl:output messageLabel="Out"
         element="dtns:Optional_Service_list "/>
      <wsdl:output messageLabel="Out"
         element="dtns:MaxID_Numeber_Class_Attributes "/>
      <wsdl:output messageLabel="Out"
         element="dtns:MaxID_Numeber_Instance_Attributes"/>
      <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
  </wsdl:operation>
```

```
<wsdl:operation name="Instance_Get_Atrribute_Single"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In"
      element="dtns:Get_Atribute_Single_Request"/>
   <wsdl:output messageLabel="Out" element="dtns:VendorID"/>
   <wsdl:output messageLabel="Out" element="dtns:DeviceType"/>
   <wsdl:output messageLabel="Out" element="dtns:ProductCode"/>
   <wsdl:output messageLabel="Out" element="dtns:Revision"/>
   <wsdl:output messageLabel="Out" element="dtns:Status"/>
   <wsdl:output messageLabel="Out" element="dtns:SerialNumber"/>
   <wsdl:output messageLabel="Out"element="dtns:ProductName"/>
   <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>

<wsdl:operation name="Class_Reset"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In" element="dtns:Reset_Request"/>
   <wsdl:output messageLabel="Out" element="dtns:Reset_Response"/>
   <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>

<wsdl:operation name="Instance_Reset"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In" element="dtns:Reset_Request"/>
   <wsdl:output messageLabel="Out" element="dtns:Reset_Response"/>
   <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>

<wsdl:operation name="Class_Get_Atrribute_All"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In"
      element="dtns:Get_Atribute_All_Request"/>
   <wsdl:output messageLabel="Out"
      element="dtns:Class_Get_Atribute_All_Response"/>
   <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>

<wsdl:operation name="Instance_Get_Atrribute_All"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In"
      element="dtns:Get_Atribute_All_Request"/>
   <wsdl:output messageLabel="Out"
      element="dtns:Instance_Get_Atribute_All_Response"/>
 <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>
<wsdl:operation name="Class_Find_Next_Object_Instance"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In"
      element="dtns:Find_Next_Object_Instance_Request"/>
   <wsdl:output messageLabel="Out"
      element="dtns:Find_Next_Object_Instance_Response"/>
   <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>

<wsdl:operation name="Instance_Set_Attribute_Single"
 pattern="http://www.w3.org/ns/wsdl/in-out">
   <wsdl:input messageLabel="In"
      element="dtns:Set_Attribute_Single_Request"/>
   <wsdl:output messageLabel="Out"
      element="dtns:Set_Attribute_Single_Response"/>
   <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
</wsdl:operation>
```

```
    <wsdl:operation name="Class_Get_Member"
     pattern="http://www.w3.org/ns/wsdl/in-out">
       <wsdl:input messageLabel="In" element="dtns:Get_Member_Request"/>
       <wsdl:output messageLabel="Out" element="dtns:Get_Member_Response"/>
       <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
    </wsdl:operation>

    <wsdl:operation name="Instance_Get_Member"
     pattern="http://www.w3.org/ns/wsdl/in-out">
       <wsdl:input messageLabel="In" element="dtns:Get_Member_Request"/>
       <wsdl:output messageLabel="Out" element="dtns:Get_Member_Response"/>
       <wsdl:outfault messageLabel="Out" ref="dtns:ObjectIdentityError"/>
    </wsdl:operation>

</wsdl:interface>
```

Definice vazeb objektu (TCP a CIP):

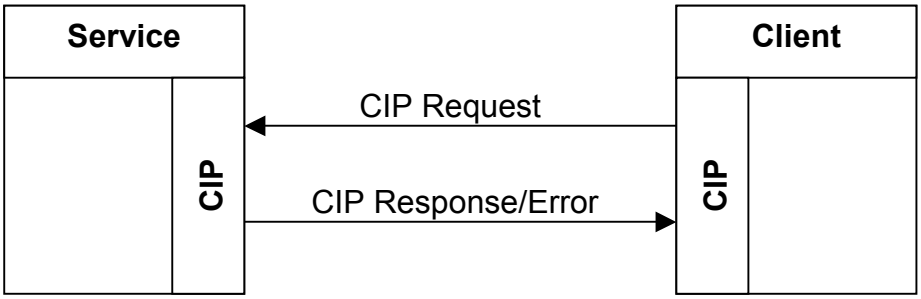
```
    <wsdl:binding name="TCPBinding"
     interface="tns: IdentityObjectInterface">
       <wsdl:operation ref="tns:Class_Get_Atrribute_Single"/>
       <wsdl:operation ref="tns:Class_Reset"/>
       <wsdl:operation ref="tns:Class_Get_Atrribute_All"/>
       <wsdl:operation ref="tns:Class_Find_Next_Object_Instance"/>
       <wsdl:operation ref="tns:Class_Get_Member"/>
       <wsdl:operation ref="tns:Instance_Get_Atrribute_Single"/>
       <wsdl:operation ref="tns:Instance_Reset"/>
       <wsdl:operation ref="tns:Instance_Get_Atrribute_All"/>
       <wsdl:operation ref="tns:Instance_Set_Attribute_Single"/>
       <wsdl:operation ref="tns:Instance_Get_Member"/>
    </wsdl:binding>

    <wsdl:binding name="CIPBinding"
     interface="tns: IdentityObjectInterface">
       <wsdl:operation ref="tns:Class_Get_Atrribute_Single"/>
       <wsdl:operation ref="tns:Class_Reset"/>
       <wsdl:operation ref="tns:Class_Get_Atrribute_All"/>
       <wsdl:operation ref="tns:Class_Find_Next_Object_Instance"/>
       <wsdl:operation ref="tns:Class_Get_Member"/>
       <wsdl:operation ref="tns:Instance_Get_Atrribute_Single"/>
       <wsdl:operation ref="tns:Instance_Reset"/>
       <wsdl:operation ref="tns:Instance_Get_Atrribute_All"/>
       <wsdl:operation ref="tns:Instance_Set_Attribute_Single"/>
       <wsdl:operation ref="tns:Instance_Get_Member"/>
    </wsdl:binding>

    <wsdl:service name="IdentityObject"
     interface="tns: IdentityObjectInterface">
       <wsdl:endpoint name="IdentityObjectEndPoint"
       binding="tns:TCPBinding"
       address="tcp://localhost/IdentityObject"/>
       <wsdl:endpoint name="IdentityObjectEndPoint"
       binding="tns:CIPBinding"
       address="cip://localhost/IdentityObject"/>
    </wsdl:service>

</wsdl:description>
```

5.2 Návrh komunikace mezi systémy

I když cílem práce není definovat úplný popis včetně vazeb na konkrétní protokoly, jsou v této kapitole naznačeny varianty, kterými může klient komunikovat s konkrétní CIP službou popsanou WSDL dokumentem.

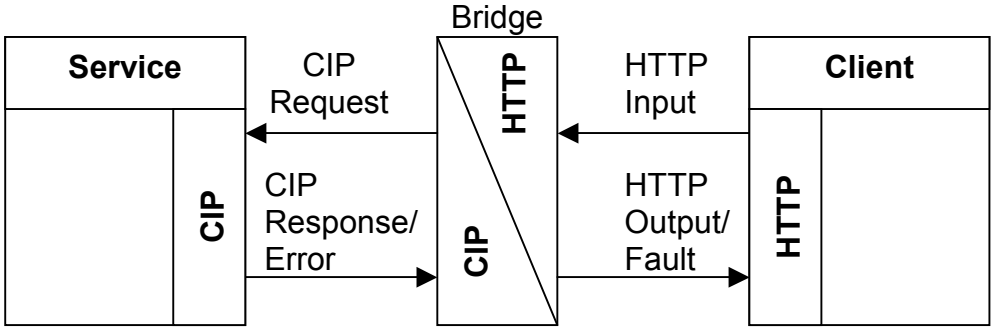
Použitím popisu služeb pomocí WSDL z předchozí kapitoly je již možné na danou službu protokolu CIP nahlížet, jako na běžnou síťovou službu srozumitelnou použitému IT rozhraní podnikového systému. Nicméně je v tomto bodě nutné ještě vyřešit fyzickou komunikaci obou systémů. Pro řešení tohoto problému jsou navrženy dvě možné varianty. Obě varianty předpokládají, že klientem (*angl. Client*) je v tomto případě podnikový systém a poskytovatelem služeb (*angl. Service*) například programovatelný automat využívající protokol CIP.

V první variantě (viz. *Obrázek 15*) je možné použít a ve WSDL dokumentu přímo specifikovat vazbu na protokol CIP. Toto je tedy řešení které musí být implementováno na straně klienta. Z tohoto důvodu je toto řešení obecně nevhodné protože systém tak ztrácí svou univerzálnost, protože každý další klient, který bude vyžadovat danou CIP službu bude již předem nucen implementovat toto rozhraní. Ovšem za předpokladu, že daný klient využívá výhradně služeb dostupných přes protokol CIP, odpadá v tomto případě nutnost implementovat další prvek v komunikační cestě mezi oběma systémy.



Obrázek 15: Varianta komunikace přes CIP protokol

Jako vhodnější varianta komunikace se ve většině případů jeví řešení znázorněné na dalším obrázku (viz. *Obrázek 16*). V této variantě je použit můstek (*angl. Bridge*) neboli prvek, který zprostředkovává převod komunikace mezi dvěma protokoly. V tomto případě tedy mezi protokolem CIP a protokolem HTTP. Při použití tohoto řešení je již klient naprosto nezávislý na protokolu CIP a komunikuje (posílá požadavky a přijímá odpovědi) s danou službou pomocí běžného a standardního protokolu HTTP.



Obrázek 16: Varianta komunikace s komunikačním mostem mezi protokoly HTTP a CIP

6 Závěr

Cílem této práce byl návrh společného popisu služeb průmyslového řídicího systému a podnikového informačního systému. Důvody a motivace pro řešení této úlohy byly nastíněny v první kapitole této práce.

V následujících dvou kapitolách práce byly popsány použité standardy a jejich modely služeb, prezentované ve formě UML diagramů. Pro názornost byly popisy standardů doplněny o konkrétní ukázky jejich použití. Pro tuto práci byly vybrány standardy CIP a WSDL 2.0. Důvody pro volbu těchto standardů jsou prezentovány v příslušných kapitolách. Hlavním důvodem pro jejich volbu byl především fakt, že se jedná o otevřené a v dnešní době hojně využívané standardy průmyslové automatizace a podnikové datové infrastruktury.

Ve čtvrté kapitole se práce zabývala možností společného popisu a mapování modelů služeb obou standardů, tedy stěžejním bodem této práce. V práci jsou navrženy dva možné přístupy k mapování. Jednotlivé varianty byly důkladně popsány a na závěr zhodnoceny jejich výhody, nevýhody a možnosti jejich použití. V samostatné podkapitole je popsáno mapování datových typů mezi dvěma použitými standardy.

V poslední kapitole je na konkrétním příkladě jednoho vybraného objektu specifikace CIP ukázáno, jak lze služby standardu CIP popsat pomocí WSDL dokumentu. Pro tento objekt jsou ukázány a podrobně popsány WSDL dokumenty obou navržených variant.

Tímto byly splněny všechny body zadání diplomové práce. Výsledky této práce je možné použít jako základ pro reálný systém, ve kterém bude možné z prostředí podnikového informačního systému ovládat a přímo přistupovat k prvkům průmyslové automatizace a tím takové prvky zakomponovat a sjednotit do jednoho celku – od manažerských po řídicí požadavky na výrobu. Možnosti jak s nejnižší vrstvou výrobního procesu komunikovat

z prostředí podnikového IT systému jsou nastíněny v závěrečné kapitole této práce. Konkrétní ukázkou řešení tohoto problému se měl zabývat poslední, nepovinný bod této práce, který nemohl být z časových důvodů splněn a měl by se tak stát prvním krokem navazujícím na poznatky a fakta uvedená v této diplomové práci.

Literatura a odkazy

[1] WSDL http://www.w3.org/TR/wsdl20-primer/

[2] THE CIP NETWORKS LIBRARY Volume 1 Common Industrial Protocol (CIP™) Edition 3.3 November, 2007 ODVA & ControlNet International Ltd.

[3] XML http://www.w3.org/TR/2006/REC-xml-20060816/

[4] W3C http://en.wikipedia.org/wiki/World_Wide_Web_Consortium

[5] *CORBA* http://en.wikipedia.org/wiki/CORBA

[6] COM http://en.wikipedia.org/wiki/Component_Object_Model

[7] DCE http://linas.org/linux/corba.html

[8] *MIDL* http://msdn.microsoft.com/en-us/library/aa367091(VS.85).aspx

[9] RPC http://en.wikipedia.org/wiki/Remote_procedure_call

[10] HTTP http://www.w3.org/Protocols/

[11] SOAP http://www.w3.org/TR/soap/

[12] *UDDI* http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[13] UML http://cs.wikipedia.org/wiki/UML

[14] Profibus http://cs.wikipedia.org/wiki/Profibus

[15] Interbus http://automatizace.hw.cz/mereni-a-regulace/ART144-uvod-do-interbus-fieldbus.html

[16] CANopen http://en.wikipedia.org/wiki/CANopen

[17] Oracle http://en.wikipedia.org/wiki/Oracle_Database

[18] SAP http://www.microsoft.com/cze/windowsserversystem/sql/solutions/sap

[19] Visual Studio http://www.microsoft.com/cze/msdn/produkty/vstudio

[20] NetBeans http://en.wikipedia.org/wiki/NetBeans

[21] Eclipse http://en.wikipedia.org/wiki/Eclipse_(software)

[22] CAN http://en.wikipedia.org/wiki/Controller_Area_Network

[23] Ethernet http://cs.wikipedia.org/wiki/Ethernet

[24] IEEE 754 http://en.wikipedia.org/wiki/IEEE_754

[25] Keeping Track of Your Production Throughout Your Enterprise Publication GMSE01-AP003A-EN-P — April 2004 Rockwell Automation

[26] Web Services (Webové služby) Petr Brňák SilverStream Software

[27] What's New in WSDL 2.0 http://www.xml.com/pub/a/ws/2004/05/19/wsdl2.html

[28] .NET Web Services Concepts

http://www.codeproject.com/KB/XML/DotNetWebServiceConcepts.aspx

[29] ISO-OSI

http://cs.wikipedia.org/wiki/Referen%C4%8Dn%C3%AD_model_ISO/OSI[30] XML

Schema http://www.w3.org/XML/Schema

[31] ODVA http://www.odva.org/

[32] Stručný popis UML http://web.sks.cz/users/ku/PRI/uml.htm

Příloha A – Stručný popis UML

Pro snazší orientaci v UML diagramech obsažených v této práci je v této příloze uveden stručný přehled tohoto modelovacího jazyka převzatý z [32].

Jednotný modelovací jazyk (*angl. Unified Modeling Language, dále jen UML*) je objektově orientovaný modelovací jazyk. Je standardizovaným jazykem pro záznam, vizualizaci a dokumentaci artefaktů systémů s převážně softwarovou charakteristikou. UML umožňuje modelovat: objekty (entity), třídy, atributy, operace (funkce) a vzájemné vztahy mezi nimi.

Základní součásti UML je notace UML (syntaxe), metamodel UML (sémantika), OCL - Object Constraint Language, specifikace převodu do výměnných formátů (CORBA IDL, XML DTD)

Notace UML – typy diagramů

1. Diagramy struktury
   - **diagram tříd** (*angl. Class diagram*)
     model statické struktury systému
   - **objektový diagram** (*angl. Object diagram*)
   - **diagram komponent** (*angl. Component diagram*)
     model komponent a jejich spolupráce
   - **diagram nasazení** (*angl. Deployment diagram*)
     model rozložení komponent při běhu systému
   - **diagram vnitřní struktury** (*angl. Composite structure diagram*)
   - **diagram balíčků** (*angl. Package diagram*)

2. Diagramy chování a interakce
   - **use case diagram**
     model funkcionality systému z pohledu uživatele
   - **sekvenční diagram** (sequence diagram)
     model časové dynamiky uvnitř Use Case - časová posloupnost zasílání zpráv mezi objekty
   - **diagram aktivit** (activity diagram)
     model průběhu jednotlivých Use Case a operací v třídách
   - **diagram komunikace**
     model komunikace spolupracujících objektů
   - **stavový diagram**
     model životního cyklu objektu - stavy objektu a přechody mezi nimi

Doplňky UML notace
stereotyp - doplnění dalšího významu standardnímu elementu
package - logická skupina elementů modelu

Pro tuto práci je důležitý především **diagram tříd**, který je dále podrobněji popsán.

Diagram tříd

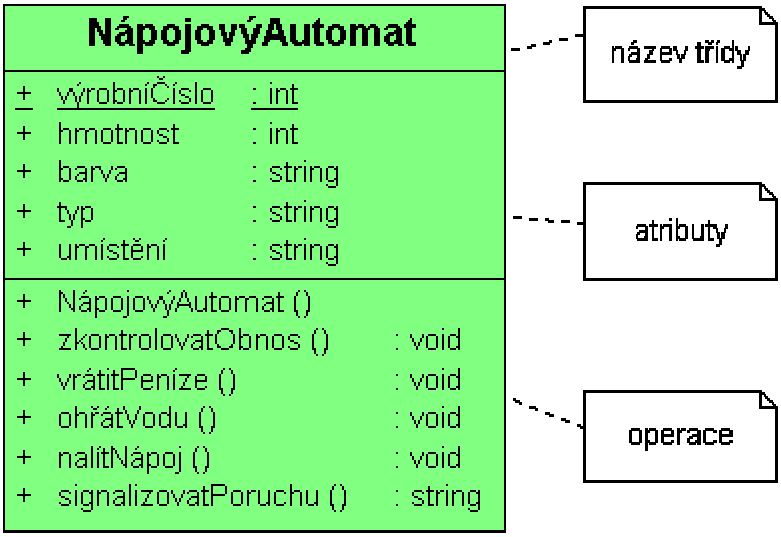
Zobrazení statické struktury systému prostřednictvím tříd a vztahů mezi nimi. Diagram tříd se využívá pro:

- fáze analýzy (konceptuální model)
- fáze návrhu (návrh atributů a operací)
- fáze implementace (návrh a tvorba programového kódu) projektovaného systému

Základní konstrukty diagramu tříd jsou:

1. Třída (*angl. Class*)

Třída sdružuje objekty se společnými vlastnostmi a chováním (sdílejí stejné atributy, operace, vztahy a sémantiku)



atribut -charakterizuje vlastnost objektu, vypovídá o stavu objektu, případně popisuje historii jeho stavů (aktivit)

operace (metoda) - funkční složka objektu, která zajišťuje jeho chování; má formu funkce nebo procedury, například:

- constructor: vytvoření instance třídy, tj. objektu (CREATE)
- get: čtení hodnoty atributu (READ)
- set: editace hodnoty atributu (UPDATE)
- destructor: zničení objektu (DELETE)

Dostupnost (*angl. visibility*) objektů diagramu tříd. Určuje, zda a jak mohou ostatní elementy modelu využívat atributy a operace daného objektu.

- private (soukromý)	pouze pro danou třídu
# protected (chráněný)	pouze pro danou třídu a její instance (dětské třídy nebo objekty)
* package	pouze pro objekty obsažené ve stejném balíčku (package)
+ public (veřejný)	pro všechny objekty

Typy tříd

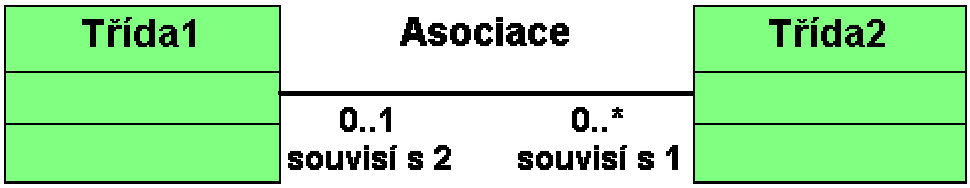
Model	*entitní, konceptuální, aplikační třída*	objekty reprezentované třídou obsahují a zpracovávají data
View	*boundary, třída uživatelského rozhraní*	objekty reprezentované třídou používají aktoři k interakci se systémem třída má definované pouze operace: s entitními třídami (jež poskytují data) je spojena asociacemi (agregacemi)
Controller	*ovládací třída*	objekty reprezentované třídou koordinují chování v systému (definují způsob reakce uživatelského rozhraní na uživatelský vstup)

2. Vztahy

Typy vztahů v UML diagramu tříd jsou následující:

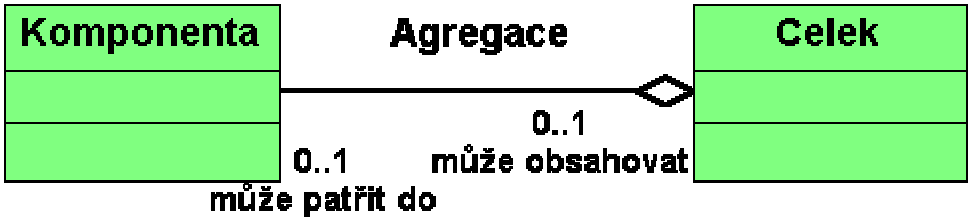
Asociace

Obecný sémantický vztah mezi prvky modelu, který specifikuje spojení mezi jejich instancemi (objekty, které vzniknou z tříd spojených asociací, si budou moci posílat zprávy).



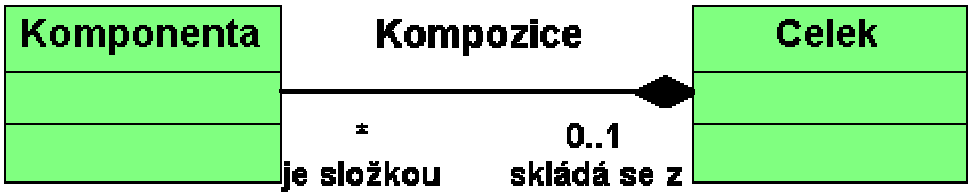
Agregace

Forma asociace, jež vyjadřuje vztah celek - část element může „přežít“ svůj kontejner, příp. stát se součástí jiného kontejneru.



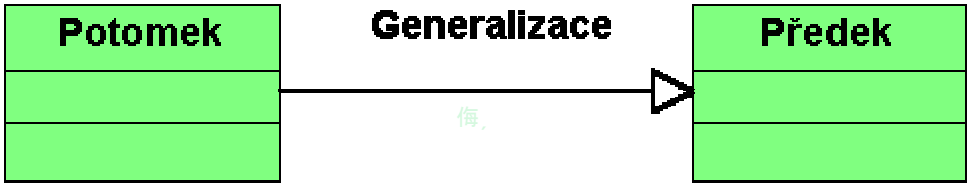
Kompozice

Silnější vazba než agregace - zrušením kontejneru automaticky zrušíme i obsažený element; daný element může být součástí právě jednoho kontejneru.



Dědičnost (generalizace)

Hierarchický vztah tříd, v němž třída - potomek (angl. subclass, specifický typ) dědí atributy a operace třídy - předka (angl. superclass, nadtyp); kromě zděděných charakteristik může mít potomek ještě své specifické vlastnosti; zděděné vlastnosti mohou být v potomkovi modifikovány.

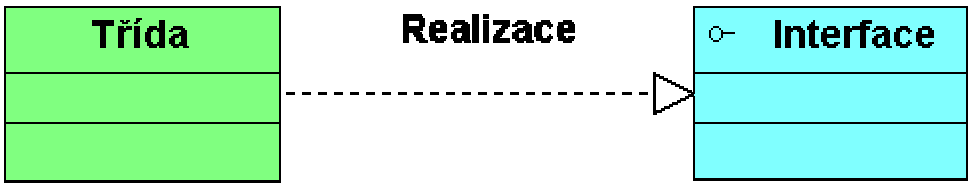


Závislost

Vztah mezi dvěma elementy modelu, v němž změna jednoho (nezávislého) elementu ovlivní druhý (závislý) element.

Realizace

Souhrn všech veřejně přístupných metod dané třídy; umožňuje vícenásobné využití operací, aniž bychom museli zavádět dědičnost mezi třídami.



Násobnost (multiplicita, kardinalita) vztahů

zápis		**význam**
0..*	*	0 až více
0..1		0 nebo 1
1..*		1 až více
1..1	1	právě 1
1..3	1,2,3	1 až 3
5,10,20		5, 10 nebo 20
5..*		5 až více

Vlastnosti správného diagramu tříd:

- je normalizovaný
- umožňuje flexibilitu
- umožňuje znuvupoužitelnost

Příloha B – Obsah přiloženého CD

- elektronická podoba této práce ve formátu .pdf
- zdrojové kódy WSDL dokumentů:
    - New_Car_Order.wsdl – ukázkový příklad použití WSDL

    Varianta mapování I:

    - Identity_Object_Class.wsdl – WSDL dokument popisující třídu Identifikačního objektu
    - Identity_Object_Instance.wsdl – WSDL dokument popisující instanci Identifikačního objektu

    Varianta mapování II:

    - Identity_Object.wsdl – WSDL dokument popisující Identifikační objekt