

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd  
Katedra informatiky a výpočetní techniky

## **Diplomová práce**

# **Určování nahraditelnosti a kompatibility webových služeba**

Místo této strany bude  
zadání práce.

# Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 27. března 2020

Zdeněk Valeš

## **Abstract**

The text of the abstract (in English). It contains the English translation of the thesis title and a short description of the thesis.

## **Abstrakt**

Text abstraktu (česky). Obsahuje krátkou anotaci (cca 10 řádek) v češtině. Budete ji potřebovat i při vyplňování údajů o bakalářské práci ve STAGu. Český i anglický abstrakt by měly být na stejné stránce a měly by si obsahem co možná nejvíce odpovídat (samozřejmě není možný doslovný překlad!).

# Obsah

<b>1</b>	<b>Úvod</b>	<b>6</b>
<b>2</b>	<b>Principy webových služeb, techniky</b>	<b>7</b>
<b>3</b>	<b>Datové typy a porovnávání</b>	<b>8</b>
3.0.1	Porovnávání datových typů . . . . .	8
<b>4</b>	<b>Popis ukládání metadat v CRCE, popis indexování API</b>	<b>9</b>
4.1	Metadata v CRCE . . . . .	9
4.2	Indexování API . . . . .	9
4.2.1	Indexování REST API . . . . .	10
4.2.2	Indexování WS . . . . .	11
4.2.3	Limity indexování . . . . .	11
<b>5</b>	<b>Popis funkce porovnávače (co se jak porovnává pro jaké typy API)</b>	<b>12</b>
5.1	Popis porovnávacího algoritmu . . . . .	12
5.1.1	Verze REST API v cestě k endpointu . . . . .	12
5.1.2	MOV flag . . . . .	12
5.2	Výsledek porovnání - Diff . . . . .	13
5.2.1	Vyhodnocení výsledku . . . . .	13
<b>6</b>	<b>Implementační detaily (jen stručně)</b>	<b>15</b>
<b>7</b>	<b>Testování</b>	<b>16</b>
	<b>Literatura</b>	<b>17</b>

# 1 Úvod

- k čemu je práce dobrá - co text práce obsahuje - use case

## 2 Principy webových služeb, techniky

- co je to API - co jsou to webové služby - REST

## 3 Datové typy a porovnávání

- přednášky z FJP - jak jazyky řeší datové typy - rekurzivní vs. nerekurzivní
- primitivní typy (v xsd) - built-in typy (v Java) - tady budu citovat [1] -
- subtyping:  $A <: B \iff A$  může být použito v kdekoliv kde je očekáváno  $B$
- kontravariance:  $F'(A) <: F(B) \iff B <: A$

### 3.0.1 Porovnávání datových typů

- jak to funguje - problémy při porovnání - subtyping vs. matching ([1])

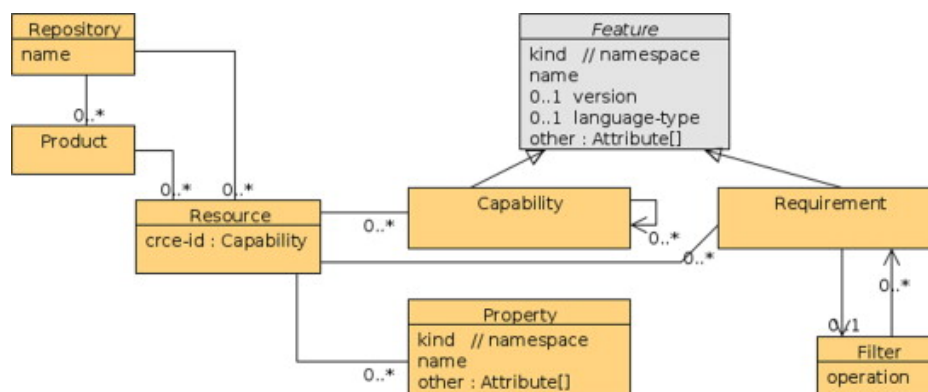


## 4 Popis ukládání metadat v CRCE, popis indexování API

- V této kapitole jsou popsány obecné způsoby ukládání metadat v CRCE - také jsou popsány podporované formáty API a způsoby jejich indexování

### 4.1 Metadata v CRCE

- tady budu citovat [2] - Resource + Capability + Properties + Atributy - stromová struktura - taky Requirements, ale ty v práci nepoužívám - Capability je rekurzivní + má namespace - Jsou root capability (přiřazené přímo Resource) a child capabilities - Capability mají Atributy + Properties - Properties mají atributy - Lze tak modelovat různé vlastnosti indexovaného objektu (viz [2], tam je to dobře popsáno)

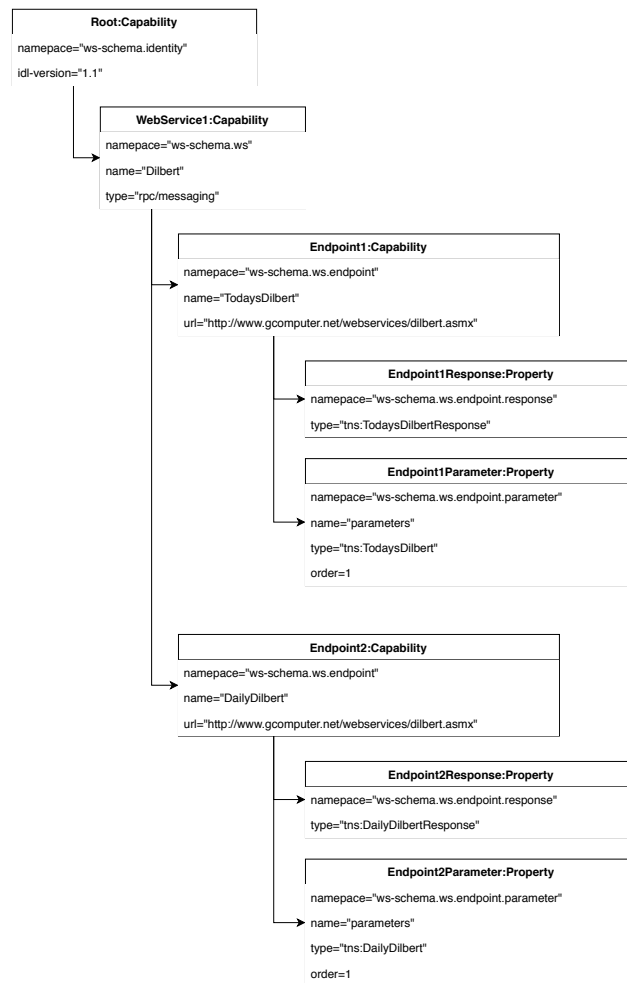


Obrázek 4.1: Reprezentace metadat v CRCE

### 4.2 Indexování API

- různé druhy jsou jinak indexované - každý druh API indexován vlastním modulem - diplomky Pejřimovského [?] a Hessové [?] - někde by asi bylo fajn ustanovit názvosloví použité v práci: - co je API: interface přístupné skrze síť (internet) - co je web service: service popsáný WSDL, WADL, nebo Json-WSP dokumentem - co je service: Service element in WSDL - co je

endpoint - WSDL: port+operation - endpoint: REST, WADL, JSON-WSP  
 - API je v CRCE uloženo jako Resource - popis API je reprezentován jako samostatná feature (1 root Capability) daného Resource - Service a endpoint jsou reprezentovány jako Capability - endpoint parametry, endpoint response, endpoint request body a endpoint request body jako Property - vlastní hodnoty pak jako Attribute - příklad metadat indexovaného API: 4.2



Obrázek 4.2: Příklad indexované SOAP web service Dilbert

### 4.2.1 Indexování REST API

- práce: [?] - binární analýza JAR s implementací API - funguje na principu hledání patternů v byte kódu - indexer vytváří hierarchii metadat ve formátu root capability -> child endpoint capabilities - podpora formátů: - REST: JAX-RS, Spring Web MVC podporovány

### 4.2.2 Indexování WS

- práce: Pejřimovského [?] - nějaký trefný obrázek indexovaných dat - konkrétní formát API detekován z buď z formátu vstupního souboru, nebo z metadata v top elementech - podle typu je pak použit daný parser - podpora formátů: - WSDL: hierarchie root capability -> web service capabilities -> child endpoint capabilities - WADL, Json-WSP: hierarchie root capability -> child endpoint capabilities - parsování souboru s popisem API, CRCE stačí i URL - indexer obsahoval drobné chyby, které jsem v rámci DP opravil - špatná indexace URL v případě WSDL (nebyla podle specifikace)

### 4.2.3 Limity indexování

- custom datové typy - 2 problémy - rekurzivní typy - jsou způsoby pro jejich rozvoj: [1] a ukládání - nicméně indexovací logika není implementovaná (ani v jednom ze zmíněných indexerů) - chybějící definice custom typů - v případě např. REST jsou uloženy v implementaci (nemusí se jednat ani o stejnou knihovnu) a indexer k nim nemusí mít přístup - tím pádem je jméno datového typu (např. fully qualified name v případě Java třídy) jedinou informací, která je o typu dostupná

## 5 Popis funkce porovnávače (co se jak porovnává pro jaké typy API)

- zmínit taky omezení, která plynou z indexovaných dat - v podstatě se porovnávají stromy Capabilit - detaily v euromicro článku

### 5.1 Popis porovnávacího algoritmu

- WSDL porovnávač má v nejhorším možném případě složitost  $O(n^3)$ , závisí na počtu WS, a počtu endpointů ve WS - problémy řešené v algoritmu: 1. jak vybrat který endpoint/ws porovnat s kterým 2. MOV - pick the best 3. datové typy (java built-in, xsd, custom) 4. kontravariance - není to úplně problém, ale při vyhodnocování finálního Diffu pro endpoint je potřeba brát v potaz - GEN/SPE může vzniknout jen z datových typů parametrů/response endpointu, takže je to poměrně přímočaré 5. verze v URL u REST API (taký vede na MOV)

#### 5.1.1 Verze REST API v cestě k endpointu

- proč: klient může chtít volat novou verzi API a je tedy žádané zjistit, jak moc je API kompatibilní (např. se mohla změnit jen implementace, takže signatura je stále stejná) - normálně by algoritmus skončil MUT, protože by kvůli rozdílným cestám k endpointům vyhodnotil endpointy z API 1 jako DEL a endpointy z API 2 jako INS - to není žádané, takže algoritmus je schopný detekovat verzi v cestě k endpointu a při výběru endpointů k porovnání ji ignorovat - podporovaný formát - `v<major>[.minor[.micro]]` - lowercase i uppercase - oddělovat může být '.', nebo '-' - regex: `\[/[vV][0-9]+(?:\.[0-9]+){0,2}`  
- pokud je detekce zapnutá, algoritmus se před porovnáním cest pokusí najít verzi a pokud ji najde, z cesty ji vyřadí a porovná cesty bez verze

#### 5.1.2 MOV flag

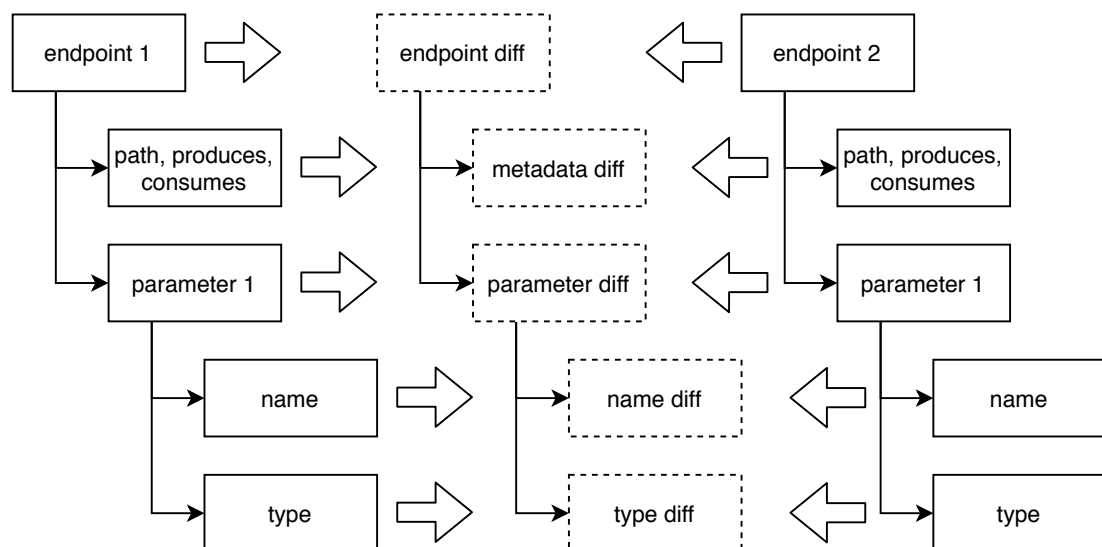
- popsát MOV - co: Příznak označující, že API/endpoint má (částečně) shodnou implementaci, ale nachází se na jiné adrese - proč: endpointy v API mohou mít jiné url/jména, ale implementačně mohou být shodné ->

potřeba detekovat - jak: na základě ostatních metadat (počet parametrů, počet endpointů ve WS)

- mov se také nastaví pokud je zaplé ignorování verzi REST API v endpoint path a pokud - cesty s verzí nejsou stejné - cesty bez verze jsou stejné
- nemusí vždy fungovat - algorimuts: - obecná detekce před samotným porovnáním -> MovDetectionResult - 3x diff: host, path to endpoint, operace
- MovDetectionResult se pak použije při výběru endpointu k porovnání a při samotném porovnání (pickBest)
- kombinace které vedou na mov: -  $h \wedge !pe \wedge !o$  -  $h \wedge pe \wedge !o$  - todo - todo

## 5.2 Výsledek porovnání - Diff

- popis výsledné datové struktury - Diff, Compatibility - vychází z [3] - stromová struktura rozdílů mezi jednotlivými uzly stromu metadat - obrázek 5.1 hezky popisuje jak to vznikne - výsledné hodnoty diffu a jejich významy pro klienta v tabulce 5.1 - SPE/GEN může vzniknout jen z daových typů parametrů/response -> lze spolehlivě použít kontravarianci a výsledek obrátit - pokud tedy vyjde SPE, znamená to např generalizovaný parametr a tedy je to pro klienta bezpečné



Obrázek 5.1: Vytvoření diffů

### 5.2.1 Vyhodnocení výsledku

- jak probíhá vyhodnocení (nejdříve se určí hodnoty listů, z nich se pak počítá dál nahoru)

Difference type	Impact on client
None (NON)	safe
Specialization (SPE)	safe
Insertion (INS)	safe
Deletion (DEL)	potentially dangerous
Generalization (GEN)	potentially dangerous
Mutation (MUT)	dangerous
Unkown (UNK)	dangerous

Tabulka 5.1: Types of differences between two nodes

## 6 Implementační detaily (jen stručně)

- zmínit, proč třídy pro porovnávání REST API a WS nemají společného předka (krom rozhraní) - důvod: chtěl jsem nechat implementaci obou porovnávačů oddělenou pro případ, že by se změnila funkce indexerů

# 7 Testování

- nějaká reálná data - STAG (WSDL) - i syntetická data



# Literatura

- [1] ABADI, M. – CARDELLI, L. On Subtyping and Matching. In *European Conference on Object-Oriented Programming (ECOOP), Lecture Notes in Computer Science*, 952, s. 145–167. ACM Press, January 1995. Dostupné z: <https://www.microsoft.com/en-us/research/publication/on-subtyping-and-matching/>.
- [2] BRADA, P. – JEZEK, K. Repository and Meta-Data Design for Efficient Component Consistency Verification. *Science of Computer Programming*. 2015, 97, part 3, s. 349–365. ISSN 0167-6423. doi: 10.1016/j.scico.2014.06.013. Dostupné z: <http://www.sciencedirect.com/science/article/pii/S0167642314002925>.
- [3] BRADA, P. – VALENTA, L. Practical Verification of Component Substitutability Using Subtype Relation. s. 38 – 45, 10 2006. doi: 10.1109/EUROMICRO.2006.50.
- [5] ]hessova2015rest HESSOVÁ, G. Automatické získání historických údajů z webových zdrojů [online]. Bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2015 [cit. 2020-02-22]. Dostupné z: <https://theses.cz/id/pzbgj7/>.
- [5] ]pejrimovsky2015ws PEJŘIMOVSÝ, D. Vytváření a ukládání popisu webových služeb v úložišti CRCE [online]. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2015 [cit. 2020-02-22]. Dostupné z: <https://theses.cz/id/bb74eq/>.