|  |  |
| --- | --- |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | |
| NÁVRHOVÉ VZORY PRE ARCHITEKTÚRY ORIENTOVANÉ NA SLUŽBY  **Diplomová práca** | |
| **2012** | **Bc. Valentín Seregély** |

|  |  |  |
| --- | --- | --- |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | | |
| NÁVRHOVÉ VZORY PRE ARCHITEKTÚRY ORIENTOVANÉ NA SLUŽBY  Diplomová práca | | |
|  | | |
| Študijný program: | | Informatika |
| Študijný odbor: | | Číslo a názov odboru (pozri zadávací list) |
| Školiace pracovisko: | | Názov katedry (Skratka katedry) |
| Školiteľ: | | Titul Meno Priezvisko, Titul |
| Konzultant: | | Titul Meno1 Priezvisko1, Titul  Titul Meno2 Priezvisko2, Titul |
| **2012 Košice** | **Valentín, Seregély Bc.** | |

|  |
| --- |
| **Abstrakt v SJ**  Abstrakt v slovenčine (referát) je povinnou súčasťou každej práce. Je výstižnou krátkou charakteristikou obsahu dokumentu. Abstrakt býva informatívny a zachováva tematické a štýlové vlastnosti práce. Nevyjadruje hodnotiace stanovisko autora. Obsahuje údaje o cieľoch práce, metódach, výsledkoch a záveroch. Text abstraktu sa píše ako jeden odstavec. Abstrakt neobsahuje odkazy na samotný text práce. Mal by mať rozsah asi 250 slov, nemal by presiahnuť jednu stranu. Pri štylizácii sa používajú celé vety, slovesá v činnom rode a tretej osobe. Používa sa odborná terminológia, menej zvyčajné termíny, skratky a symboly sa pri prvom výskyte v texte definujú.  **Kľúčové slová**  Kľúčové slovo1, kľúčové slovo2, ... |

|  |
| --- |
| **Abstrakt v AJ**  Text abstraktu v svetovom jazyku je potrebný pre integráciu do medzinárodných informačných systémov (napr. The Network Digital Library of Theses and Dissertations). Ak nie je možné jazykovú verziu umiestniť na jednej strane so slovenským abstraktom, je potrebné umiestniť ju na samostatnú stranu (cudzojazyčný abstrakt nemožno deliť a uvádzať na dvoch stranách).  **Kľúčové slová v AJ**  Key word 1, Key word 2,... |

**Zadanie práce**

Tu vložte naskenované zadanie úlohy.

Odporúčame skenovať na 200-300 DPI, čierno-bielo (alebo 2 farby)

! v jednej vytlačenej ZP musí byť vložený **originál** zadávacieho listu !

**Čestné vyhlásenie**

Vyhlasujem, že som celú diplomovú prácu vypracoval/a samostatne s použitím uvedenej odbornej literatúry.

Autori metodických príručiek (pozri Katuščák , Gonda ) o záverečných prácach sa nazdávajú, že takéto vyhlásenie je zbytočné, nakoľko povinnosť vypracovať záverečnú prácu samostatne, vyplýva študentovi zo zákona a na autora práce sa vzťahuje autorský zákon.

Košice, 15. máj 2012 ..........................................

*vlastnoručný podpis*

**Poďakovanie**

Na tomto mieste môže byť vyjadrenie poďakovania napr. vedúcemu práce resp. konzultantom za pripomienky a odbornú pomoc pri vypracovaní práce. Nie je zvykom ďakovať za rutinnú kontrolu, menšiu spoluprácu alebo všeobecné rady. Vyjadrenie poďakovania v prípade využitia inej práce sa uskutočňuje formou citácie na konci hlavného textu práce a odkazy na citáciu sa musia uviesť aj na zodpovedajúcich miestach v texte.

**Predhovor**

Predhovor je povinnou náležitosťou diplomovej práce (pozri Gonda (). V predhovore diplomant uvedie základné charakteristiky svojej diplomovej práce a okolnosti jej vzniku. Vysvetlí dôvody, ktoré ho viedli k voľbe témy, cieľ a účel práce a stručne informuje o hlavných metódach, ktoré pri spracovaní diplomovej práce použil. Napr.:

Problematika .... je predmetom početných výskumov. U nás sa týmito otázkami zaoberá XY v učebnici Názov učebnice. Vo svete ...

Hlavný dôvod, ktorý ma viedol k výberu témy bol... Zámerom mojej práce je ... Mojou ambíciou je vyriešiť ...

Predkladaná diplomová práca vznikla za spolupráce ...

**Obsah**

[Zoznam obrázkov 9](#_Toc353631312)

[Zoznam tabuliek 10](#_Toc353631313)

[Zoznam symbolov a skratiek 11](#_Toc353631314)

[Slovník termínov 12](#_Toc353631315)

[Úvod 13](#_Toc353631316)

[1 Úvod do architektúr orientovaných na služby 14](#_Toc353631317)

[1.1 Typy SOA 15](#_Toc353631318)

[1.1.1 Architektúra Služby 16](#_Toc353631319)

[1.1.2 Architektúra kompozície služieb 16](#_Toc353631320)

[1.1.3 Architektúra inventára služieb 17](#_Toc353631321)

[1.1.4 Architektúra spoločnosti orientovanej na služby 17](#_Toc353631322)

[1.2 Ciele orientácie na služby 17](#_Toc353631323)

[2 Návrhové vzory 19](#_Toc353631324)

[2.1 SOA vzory 19](#_Toc353631325)

[3 Inventár 22](#_Toc353631326)

[3.1 Základné vzory inventára služieb 22](#_Toc353631327)

[3.1.1 Inventory Boundary Patterns 22](#_Toc353631328)

[3.1.2 Inventory Structure Patterns 25](#_Toc353631329)

[3.1.3 Inventory Standardization Patterns 29](#_Toc353631330)

[3.2 Vzory logických vrstiev inventára 32](#_Toc353631331)

[3.3 Vzory Centralizácie inventára 35](#_Toc353631332)

[3.4 Vzory implementácie inventára 36](#_Toc353631333)

[3.5 Vzory riadenia inventára 37](#_Toc353631334)

[4 Aplikácia vzorov 39](#_Toc353631335)

[4.1 Aplikácia vzoru Service Normalization 39](#_Toc353631336)

[4.2 Aplikácia vzoru Logic Centralization 39](#_Toc353631337)

[4.3 Aplikácia vzoru Service Layers 40](#_Toc353631338)

[4.4 Aplikácia vzoru Entity Abstraction 41](#_Toc353631339)

[4.5 Aplikácia vzoru Utility Abstraction 41](#_Toc353631340)

[4.6 Aplikácia vzoru Process Abstraction 42](#_Toc353631341)

[4.7 Aplikácia vzoru Process Centralization 44](#_Toc353631342)

[4.8 Aplikácia vzoru Metadata Centralization 45](#_Toc353631343)

[4.9 Aplikácia vzoru Inventory Endpoint 46](#_Toc353631344)

[4.10 Aplikácia vzoru Canonical Protocol 47](#_Toc353631345)

[4.11 Aplikácia vzoru Canonical Schema 47](#_Toc353631346)

[4.12 Aplikácia vzoru Canonical Expression 48](#_Toc353631347)

[5 Záver 49](#_Toc353631348)

[Zoznam použitej literatúry 50](#_Toc353631349)

[Prílohy 53](#_Toc353631350)

[Curriculum vitae 54](#_Toc353631351)

Zoznam obrázkov

Obr. 1 Enterprise Inventory 23

Obr. 3 Domain Inventory 24

Obr. 4 Service Normalization 26

Obr. 5 Logic Centralization 27

Obr. 6 Service Layers 28

Obr. 7 Canonical Protocol 30

Obr. 8 Utility Abstraction 32

Obr. 9 Entity Abstraction 33

Obr. 10 Process Abstraction 34

Zoznam tabuliek

Tab. 1 18

Tab. 2 19

Zoznam symbolov a skratiek

μ **micro**, 10-6

SI **S**ystème **I**nternational

V **volt**, základná jednotka napätia v sústave SI

Slovník termínov

**Dizertácia** je rozsiahla vedecká rozprava, v ktorej sa na základe vedeckého výskumu a s použitím (využitím) bohatého dokladového materiálu ako i vedeckých metód rieši zložitý odborný problém.

**Font** je súbor, obsahujúci predpisy na zobrazenie textu v danom písme, napr. na tlačiarni. To čo vidíme je písmo; font je súbor a nevidíme ho.

**Meter** (m) je vzdialenosť, ktorú svetlo vo vákuu prejde za časový interval 1/299 792 458 sekundy.

**Proces** je postupnosť či rad časovo usporiadaných udalostí tak, že každá predchádzajúca udalosť sa zúčastňuje na determinácii nasledujúcej udalosti.

Úvod

Hlavný text práce obsahuje úvod, jadro (číslované kapitoly a podkapitoly druhej a tretej úrovne s ilustráciami a tabuľkami), záver a zoznam použitej literatúry. Úvod nemá byť číslovaný.

Úvod stručne a jasne, pritom však podrobnejšie ako v predhovore (pozri Gonda )

* vyjadruje stav poznania alebo praxe v danej oblasti, ktorá je predmetom práce,
* zdôvodní aktuálnosť témy,
* nastolí problémy, ktoré chce vyriešiť,
* vysvetlí účel a ciele práce,
* opíše použité metódy a postup riešenia,
* uvedie vzťah práce k ďalším prácam v danej oblasti, spresní informačné zdroje a pramene, ktoré najviac využíval (a ktoré uvedie v zozname použitej literatúry),
* zdôvodní význam riešenia problematiky,
* načrtne stručný obsah kapitol.

V úvode nie je potrebné opakovať to, čo je uvedené v abstrakte. Nie je vhodné podrobne opisovať metódy, experimentálne výsledky, ani opakovať to, čo je uvedené v závere. Aj keď je úvod umiestnený na začiatku, jeho konečnú verziu píše diplomant až po dokončení celej práce.

1. Úvod do architektúr orientovaných na služby

Pretože termín servisne orientovaný už existuje niejakú dobu, bol použitý v rôznych kontextoch a pre rôzne účely. Čo sa nemení počas jeho existencie, je to, že predstavuje odlišný prístup pre oddelovanie záujmov. To znamená, že logiku potrebnú pre riešenie niejakého problému sa dá lepšie vytvoriť, uskutočniť a riadiť, ak ju rozdelíme na kolekciu menších vzájomne súvisiacich častí. Každá z týchto častí súvisí s niejakým záujmom alebo určitou časťou problému. Tento prístup prevyšuje technologické a automatizačné riešenia. Je to zavedená a všeobecná teória, ktorou sa dá riešiť rôzne problémy. To čo odlišuje servisne orientovaný prístup pre oddelenie záujmov, je spôsob, akým oddelenie dosahuje. Aby sa udržala nezávislosť, služby zapúzdrujú logiku v odlišnom kontexte. Tento kontext môže byť špecifický vzhľadom k riadiacej úlohe alebo inému logickému zoskúpeniu. Veľkosť a oblasť logiky reprezentovanou danou službou môže byť rozličný. Naviac logika služby môže zahrňovať logiku poskytovanú ďalšími službami. V takom prípade spolu jedna alebo viac služieb tvorí kolektív. Napríklad riešenie automatizácie riadenia je typickou implementáciou riadiaceho procesu. Tento proces sa skladá z logiky, ktorá určuje akcie vykonávané daným riešením. Logika je rozložená do série krokov, ktoré sa spúšťajú v dopredu definovanom poradí podla riadiacich pravidiel a podmienok spustenia. Pri riešení atomatizácie zloženého zo služieb, každá služba môže zapúzdrovať úlohu vykonávanú v jednom kroku alebo v podprocese zloženého zo skupiny krokov. Služba môže dokonca zapúzdriť celú logiku procesu. Pri posledných dvoch prípadoch smie väčšiu oblasť reprezentovanú službami zahrňovať logiku, ktorú zapúzdrujú ďalšie služby.

V SOA môžu služby používať iné služby alebo programy. Vzťah medzi službami je založený na tom, že služby o sebe musia navzájom vedieť, aby mohli spolu komunikovať. K tomuto účelu slúžia popisy služieb. Popis služby vo svojej najzákladnejšej forme, stanovuje názov, umiestnenie služby a požiadavky na výmenu dát. Spôsob akým služby používajú popisy, končia vzťahom klasifikovaným ako voľná väzba.

Aby spolu služby komunikovali a vykonali niečo zmysluplné, musia si vymeniť informácie. Je teda potrebný komunikačný systém, ktorý je schopný zachovať ich vzťah voľnej väzby. Jedným takým systémom je výmena správ.

Komunikácia služieb

Akonáhle služba odošle správu na svoju cestu, stratí kontrolu nad tím, čo sa potom správe stane. Z tohto dôvodu je potrebné, aby boli správy nezávislými jednotkami komunikácie. To znamená, že správy, rovnako ako služby, by mali byť autonómne. Aby to dosiahli, správy môžu byť vybavené dostatočnou "inteligenciou", aby samostatne spravovali svoju časť procesnej logiky. Služby, ktoré poskytujú popisy služieb a komunikujú prostredníctvom správ, tvoria základnú architektúru. Zatiaľ sa táto architektúra javí podobne ako z minulosti známe distribuované architektúry, ktoré podporujú výmenu správ a oddelujú rozhranie od procesnej logiky. To čo odlišuje túto architektúru, je to, ako sú navrhnuté tri komponenty jadra (služby, popisy a správy). To je miesto, kde na scénu vstupuje orientácia na služby.

Návrh služieb

Aplikáciou princípov orientácie na služby na procesnú logiku vzniká štandardizovaná procesná logika orientovaná na služby. Ak sa riešenie skladá z jednotiek logiky orientovanej na služby, tak sa nazýva riešením orientovaným na služby.

Implementácia SOA

Rôzné abstraktné modely SOA existovali už pred príchodom webových služieb. Avšak najvhodnejšou implementáciou služieb sú práve webové služby. V súčastnosti všetky hlavné platformy podporujú tvorbu riešení orientovaných na služby, a všetky sú založené na webových službách.

* 1. Typy SOA

Z hľadiska architektúry sa SOA delí na nasledujúce typy:

Architektúra služby – Architektúra jednej služby

Architektúra kompozície služieb – Architektúra určitej množiny služieb, ktorá tvorí kompozíciu služieb

Architektúra inventára služieb – Architektúra, ktorú tvorí kolekcia služieb, ktoré spolu súvisia. Jednotlivé služby sú riadené nezávisle od ostatných.

Architektúra spoločnosti orientovanej na služby – Architektúra spoločnosti, ktorej akákoľvek časť je servisne orientovaná.

Tieto architektúry so sebou vo veľkej miere súvisia. Architektúra služby spadá pod architektúru kompozície. Architektúra kompozícia služieb je časťou architektúry inventára služieb a všetky predchádzajúce sú súčasťou architektúry spoločnosti orientovanej na služby.

* + 1. Architektúra Služby

Architektúra služby predstavuje architektúru softvéru, ktorý je navrhnutý ako služba. Táto architektúra je z hľadiska rozsahu porovnateľná s architektúrou komponentu, avšak rozsah architektúry služby býva väčší pretože služba môže zahŕňať aj viacero komponentov. Špecifikáciu takejto architektúry obvykle vlastnia správcovia služieb. Táto špecifikácia by mala byť skrytá pred ostatnými členmi vývojárskeho týmu.

Dôležitou časťou architektúry služby je jej kontrakt, ktorý vyjadruje rozsah a vlastnosti logiky a požiadavky na implementáciu. Podľa štandardu orientácie na služby je kontrakt prvou časťou služby ktorá je doručená.

* + 1. Architektúra kompozície služieb

Hlavným cieľom doručovania skupiny nezávislých služieb je možnosť ich kombinácie do kompozície služieb. Takáto kompozícia je schopná riešiť väčšie a komplexnejšie úlohy. Každá kompozícia je postavená na svojej architektúre kompozície služieb, ktorá obsahuje architektúry všetkých zúčastnených služieb. Jedným z aspektov architektúry kompozície služieb je, že môže byť časťou väčšej rodičovskej kompozície, takže jedna kompozícia sa može skladať alebo sa odkazovať na viaceré ďalšie kompozície.

Architektúra kompozície služieb však zahrňuje viac ako len architektúry alebo kontrakty jednotlivých služieb. Kompozícia väčšinou obsahuje špecifickú službu ktorej úlohou je riadenie kompozície. Detaily takejto služby sú menej skrývané a jej konštrukcia väčšinou poskytuje veľkú časť logiky kompozície. Architektúra kompozície je vo veľkej miere závislá na prostredí v ktorom sú jej služby umiestnené. Špecifikácia tejto architektúry môže zahŕňať bezpečnosť, manažment tranzakcií, odosielanie správ a rôzne iné aspekty infraštruktúry.

* + 1. Architektúra inventára služieb

Architektúra inventára služieb je kolekcia individuálne štandardizovaných a riadených služieb, ktoré sú dodávané v rámci vopred definovaných medzí. Táto kolekcia služieb sa nešpecializuje na jediný biznis proces ale v ideálnom prípade pokrýva viacero procesov. Architektúra inventára môže mať dosah na celú spoločnosť alebo iba na niektorú doménu v rámci spoločnosti.

* + 1. Architektúra spoločnosti orientovanej na služby

Táto forma architektúry predstavuje architektúry služby, kompozície služby a inventára služieb v rámci jednej spoločnosti. Architektúra spoločnosti orientovanej na služby je porovnateľná s tradičnou architektúrou spoločnosti iba ak väčšia časť alebo celé technické prostredie je orientované na služby. V opačnom prípade ide iba o dokumentáciu jednotlivých častí spoločnosti, ktoré sú orientované na služby.

* 1. Ciele orientácie na služby

Orientácia na služby ako návrhový prístup bola vytvorená pre dosiahnutie nasledujúcich cieľov:

* Zvýšenie spolupráce – spolu súvisiace služby sú navrhnuté tak aby boli kompatibilné a aby mohli byť navzájom spájané.
* Zvýšená federácia – služby majú rovnakú vrstvu kontraktov, ktorá skrýva ich vnútorné rozdielnosti, vďaka čomu môžu byť spravované a vyvíjané individuálne.
* Väčšie možnosti diverzifikácie dodávateľov – architektúra orientovaná na služby nieje založená na žiadnom dodávateľovi, takže spoločnosť nieje z tejto stránky obmedzovaná pri vývoji architektúry.
* Väčšie prepojenie technologií a obchodu – Niektoré služby sú navrhnuté tak, že ich funkcionalita odráža a vyvíja sa spolu s obchodom danej spoločnosti
* Vyššia návratnosť investície – Od väčšiny služieb sa očakáva, že viacnásobne vrátia vynaložené prostriedky na ich dodanie a prevádzky.
* Väčšia agilita – Nové a meniace sa podnikové požiadavky sa dajú splniť rýchlejšie v prostredí v ktorom môžu byť riešenia vytvorené pomocou znovupoužiteľnosti a spolupráce už existujúcich služieb.
* Nižšie náklady na IT – IT oddelenie môže podporovať spoločnosť lepšie ak s nižšími nákladmi produkuje väčšiu hodnotu vďaka predchádzajúcim výhodám.

1. Návrhové vzory

V prvom rade je dôležité definovať, že čo sú návrhové vzory a načo sú dobré. Návrhový vzor je vlastne overené riešenie niejakého všeobecného problému pri návrhu. Mal by byť zaznamenaný v konzistentnom formáte a väčšinou je súčasťou väčšej množiny vzorov.

Výhody návrhových vzorov sú nasledovné:

* Predstavujú overené riešenie všeobecných problémov pri návrhu
* Návrh je vďaka nim štandardizovaný
* Väčšinou sú opakovateľné profesionálmi v návrhu
* Môžu byť použité pre zaistenie konzistencie návrhu a realizácie.
* Môžu sa stať základom návrhových štandardov.
* Väčšinou sú flexibilné a voliteľné.
* Môžu byť použité na študíjne účely vďaka tomu, že dokumentujú špecifické aspekty návrhu systému.
* Môžu byť použité pred implementáciou systému.
* Môžu byť podporené aplikáciou iných návrhových vzorov tej istej skupiny návrhových vzorov.
  1. SOA vzory

SOA návrhové vzory spoločne tvoria jazyk vzorov, ktorý dovoľuje použitie vzorov v rôznych kombináciách a rôznom poradí. Niektoré jazyky vzorov sú viac otvorené a dovoľujú kombináciu vzorov do rôznych sekvencií, iné sú viac štrukturované a odporúčajú poradie vzorov. Štrukturované jazyky rozdeľujú vzory podľa takzvanej zrnitorsti. Hrubšie vzory sú aplikované skôr ako tie jemnejšie, ktoré stavajú na hrubších alebo ich rozširujú.

Štrukturované jazyky vzorov majú nasledujúce výhody:

* Vedia organizovať skupiny prakticky overených návrhových vzorov do tak isto prakticky overených sekvencií.
* Zaručujú konzisteciu v spôsobe dosiahnutia návrhových cieľov
* Sú efektívnym učebným nástrojom, ktorý poskytuje náhľad do príčiny a spôsobu aplikovania určitých techník a metód.
* Sú flexibilné, pretože konečný výber sekvencie vzorov je na odborníkovy.

Ďalej existujú zložené vzory, ktoré sa skladajú z viacerých individuálnych navrhových vzorov. SOA návrhové vzory niesú viazané na žiadnú tehnologickú platformu alebo obchodné odvetvie. Sú to jednoducho techniky návrhu pre dosiahnutie určitích cieľov a benefitov spájaných s orientáciou na služby a zároveň predídeniu špecifických prekážok. Pomocou návrhových riešení a techník, použitých v návrhových vzoroch SOA, môžeme získať náhľad do mechanizmu orientácie na služby a do architektonického modelu ktorý SOA reprezentuje.

Každý z uvedených vzorov je unikátny, odlišný od ostatných a je považovaný za rovnocenného v celom katalógu vzorov, ale môže byť užitočné uviesť určité skupiny podobných vzorov pre lepšie pochopenie ich pomenovania a ich všeobecnej charakteristiky.

Canonical Patterns – je skupina vzorov, ktorá uvádza za najlepšie riešenie daného problému uvedenie návrhového štandardu. Správna aplikácia takéhoto typu vzoru vyúsťuje v kanonických konvenciách, ktoré garantujú konzistentný návrh v rôznych častiach inventára.

Centralization Patterns – Centralizácia v tomto kontexte znamená zjednodušenie všetkých možností na jednu jedinú. Aplikácia tohto konceptu v klúčových častiach servisne orientovanej architektúry zavádza konzistenciu, podporuje štandardizáciu a znovupoužitie.

V nasledujúcich kapitolách budú predstavené vybrané vzory z množiny vzorov pre inventár služieb. Každý zo vzorov bude jednoducho popísaný nasledujúcim spôsobom.

* Problém – Sekcia s vysvetlením problému, okolností, ktoré knemu vedú a následkov, ktoré z neho vyplívajú. Daný návrhový vzor navrhuje riešenie tohoto problému.
* Riešenie – Predstavuje riešenie problému navrhované daným vzorom. Obsahuje popis princípu riešenia, a v krátkosti približnú aplikáciu takéhoto riešenia.
* Následky – Väčšina vzorov nieje bez kompromisov. V tejto sekcii budú predstavené následky a náklady aplikácie daného vzoru, a požiadavky na prostredie daného vzoru.

1. Inventár
   1. Základné vzory inventára služieb

Návrhové vzory tejto skupiny vzorov riešia návrhové problémy spojené so štruktúrou architektúry, ktorá sa snaží o dosiahnutie flexibility a agilného prostredia logiky navrhutej v súlade s orientáciou na služby. Vzory tejto skupiny návrhových vzorov sú organizované v troch skupinách:

* Inventory Boundary Patterns – Do tejto podskupiny patria vzory určujúce rozsah daného inventára
* Inventory Structure Patterns – Podskupina vzorov, ktoré definujú vnútornú štruktúru daného inventára.
* Inventory Standardization Patterns – Podskupina vzorov, ktoré zavádzajú návrhové štandardy zaručujúce interoperabilitu služieb.
  + 1. Inventory Boundary Patterns

**Enterprise Inventory**

Problém

V spoločnostiach je vývoj jednotlivých služieb väčšinou rozdelený do rôznych vývojových tímov. Preto sa môže stať, že jednotlivé služby od rôznych tímov budú postavené na rôznych technologických architektúrach. Tieto rozlišnosti môžu viesť k vážnym problémom pri kompozícii služieb, pretože takéto služby môžu byť nekompatibilné.

Riešenie

Riešením takéhoto problému v podniku orientovanom na služby môže byť vytvorenie inventára služieb pre tento podnik. Tento krok by vyriešil problémy tohto typu pretože služby vyvíjané pre daný inventár služieb by museli splňať určité požiadavky, vďaka čomu by boli navzájom kompatibilné. Tento vzor je najvhodnejší pre nasledujúce typy prostredí:

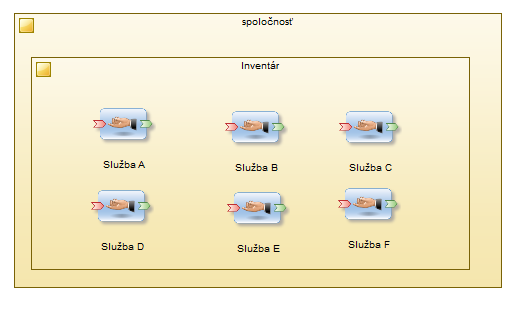
* Malé až stredne veľké organizácie s dostatkom zdrojov
* Stredné až veľké organizácie s kontrolovaným IT prostredím, zavedenými celopodnikovými štandardmi alebo spoločnosti dostatočne flexibilné pre adaptáciu požadovanej úrovne štandardizácie
* Stredné až veľké organizácie s dostatkom zdrojov k súčasnému fungovaniu existujúcich systémov a budovaniu Enterprise inventára.
* Nové organizácie bez existujúcich systémov, ktoré budujú svoje IT prostredie od začiatku.

Inventár vytvorený pomocou tohoto vzoru nemusí v každom prípade zahŕňať všetky služby danej spoločnosti, jeho zmyslom je vytvorenie inventára, ktorého rozsah je dostatočne veľký, aby ho vôbec malo zmysel vytvoriť.

Aplikácia tohoto vzoru nevedie k vytvoreniu nových služieb alebo nejakého fyzického inventára, ale k ustanoveniu konceptu inventára služieb spoločnosti, ktorého sa služby do neho patriace musia držať.

Následky

Pred nasadením inventára spoločnosti je potrebné vykonať viacero analýz celého podniku. Vďaka nim sa zaistí kompatibilita navrhovaných služieb. Avšak tieto analýzy sa môžu ľahko predražiť a predlžiť celý projekt. Alternatívnym prístupom, bez spomínaných nevýhod je vzor Domain inventory.



Obr. Enterprise Inventory

**Domain inventory**

Problém

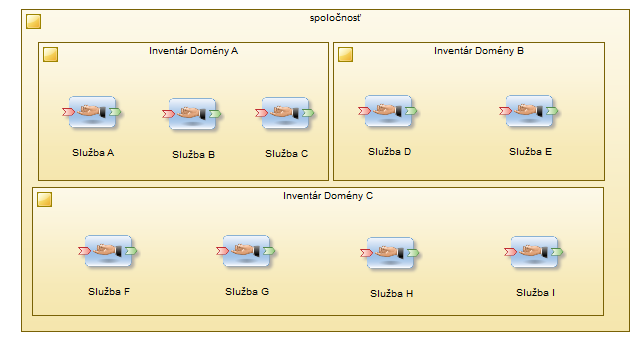
Vytvorenie a údržba jedného inventára pre celú spoločnosť je v mnohých prípadoch nemožná alebo príliž komplikovaná. V takom prípade je vhodné použitie vzoru Domain inventory, ktorý rieši ten istý problém ako Enterprise inventory bez spomínaných nevýhod.

Riešenie

Riešením je vytvorenie inventára pre každú doménu daného podniku. Pričom všetky inventáre služieb sú štandardizované a riadené nezávysle. Pri tomto vzore je najdôležitejšia jasná definícia domén podniku ešte pred vytváraním inventárov

Následky

Tento vzor dovoľuje individuálne štandardizovanie inventárov, čo často vedie k problémom v kompatibilite medzi službami z rôznych inventárov. Tento problém sa dá riešiť zavedením cielenej transformácie pri spolupráci medzi rôznymi službami. Avšak návrh a vývoj sa tým zbytočne komplikuje a tiež to predstavuje väčšiu záťaž pri behu systému. Nezávislosť inventárov prínaša aj riziko tvorby redundantných služieb.



Obr. Domain Inventory

* + 1. Inventory Structure Patterns

**Service Normalization**

Problém

Pri tvorbe inventárov viacerými vývojárskymi týmami, je vždy riziko tvorby služieb, ktorích funkcionalita sa prekrýva, aj keď sú jasne definované hranice inventárov. Tento jav vedie k denormalizácii inventára.

Riešenie

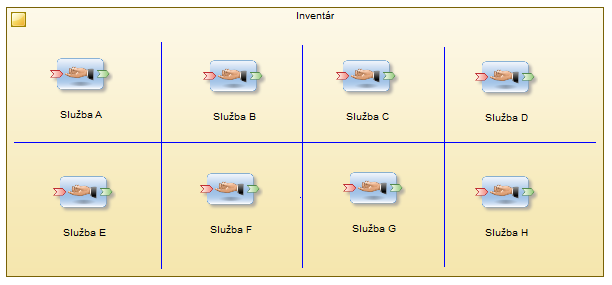
Modelovanie služieb ešte pred vytvorením ich kontraktov pomáha riešiť tento problém, pretože takto sú hranice služieb definované, čo zaisťuje, že sa nebudú prekrývať. Takto sa dá vytvoriť normalizovaný inventár služieb. Aplikácia tohoto vzoru prebieha iteratívne, v nasledujúcich krokoch:

* Identifikácia a dekompozícia biznis procesu ktorý spadá pod daný inventár
* Pridelenie jednotlivých častí procesu do nových alebo už existujúcic služieb
* Kontrola, či sa žiadne hranice služieb neprekrývajú.

Tieto kroky sú časťou procesu modelovania služieb ktorý berie do úvahy aj iné hľadiská modelovania. Pre úplné zavedenie vzoru Service Normalization je potrebné tento proces realizovať iteratívne, pričom každá iterácia predstavuje spracovanie jedného biznis procesu, ktorý spadá pod daný inventár. Vďaka týmto iteráciám sa funkčny kontext a hranice služieb opakovane vylepšujú a validujú.

Následky

Kvôli potrebe modelovania všetkých služieb môže byť pre väčšie podniky výhodnejšie spustenie projektov na analýzu, ktoré musia byť dokončené ešte pred začatím tvorby hociktorej služby. Kvôli tomuto kroku navyše sa môže projekt značne predlžiť. Ďalej je potrebná sústavná kontrola normalizácie služieb počas ich budúceho vývoja, čo môže komplikovať údržbu.



Obr. Service Normalization

**Logic Centralization**

Problém

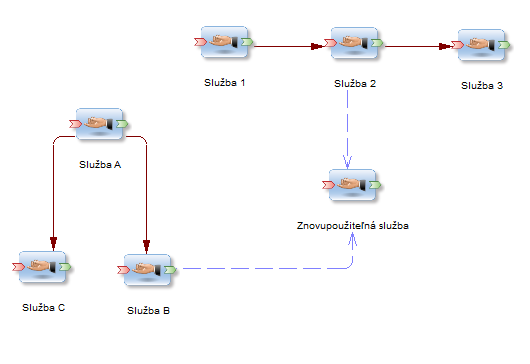
Medzi najväčšie výhody architektúry orientovanej na služby patrí možnosť znovupoužitia služieb. Avšak nič nezaručuje, že jednotlivé vývojarské týmy budú používať už existujúce služby, pretože z krátkodobého hľadiska môže byť jednoduchšie a lahšie vytvoriť služby s podobnou funkcionalitou. Takýto krok môže byť pohodlnejší ako znovupoužitie, ale jednoznačne vedie k denormalizácii inventára služieb a funkčnej redundancii.

Riešenie

Pri práci viacerých vývojárskych týmoch vždy existuje riziko tvorby už existujúcej funkcionality. Príčinou môže byť to, že nevedia o existencii danej služby, napríklad preto lebo nieje zistitelná. Alebo jednoducho považujú jej použitie za zaťažujúce. Ak je problém v jeho použití, tak je odporúčané použiť vzor Logic centralization, ktorý určuje služby ako jediné prostriedky pre prístup k oblasti funkcionality ktorá danej službe v inventáry patrí. Takže ak je potrebné použitie určitej funkcionality tak sa musí použiť služba, ktorá má túto funkcionalitu vo svojej funkčnej oblasti. V prípade, že niejaká potrebná funkcionalita ešte nieje implementovaná, treba ju implementovať do služby ktorej spadá daná funkčná oblasť. Tento vzor je aplikovaný vytvorením a vynútením dodržovania štandardov nariaďujúcich používanie služieb ako jediného vstupného bodu pre funkcionalitu ktorú ohraničujú v danom inventáry. Takýto štandard nariaďuje, že agnostické služby musia byť použité vždy ak daná funkcionalita spadá pod ich hranice, a to aj v prípade, že nespĺňajú všetky funkčné požiadavky.

Následky

Vo väčších spoločnostiach môže byť nasadenie tohoto vzoru ťažké. Hlavne ak v minulosti nebol kladený dôraz na znovupoužiteľnosť služieb a použitie návrhových štandardov. Navyše použitie tohoto vzoru môže vyvolať v takých prípadoch negatívne reakcie vývojárov.



Obr. Logic Centralization

**Service Layers**

Problém

V inventáry služieb sa väčšinou nájdu služby s podobnou funkcionalitou aj keď sú navrhované a implementované samostatne. Toto vedie k narušeniu konzistencie definovaných hraníc služieb a funkčnej redundancii.

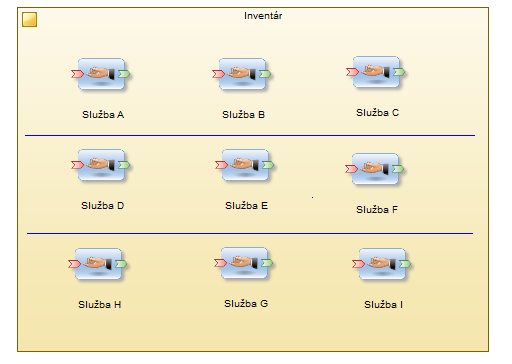
Riešenie

Riešenie môžu predstavovať modely služieb, ktoré združujú podobné služby v danom inventáry. V inventáry sa môže nachádzať viacero modelov, pričom každý reprezentuje návrhové charakteristiky, ktorých sa musia služby daného modelu držať. Služby daného modelu potom tvoria logickú vrstvu architektúry, ktorá môže byť vyvíjaná a riadená ako samostatná skupina. Vrstvy by mali byť definované ešte pred implementáciou inventára, a každý inventár by mal obsahovať minimálne dve vrstvy.

Základné vrstvy sú:

* Vrstva s jedno účelovou logikou (neagnostická)
* Vrstva s mnoho účelovou logikou (agnostická)

Abstrakciou neagnostickej logiky do jednej z častí inventára, môže byť agnostická logika definovaná a vyvíjajúca sa na podporu znovupoužiteľnosti. Aj keď stále ostáva možnosť pre upravovanie vrstiev služieb, najbezpečnejšie je zakladať jednotlivé vrstvy na bežných modeloch danej oblasti. Tieto základné modely sú overené v riešení známych návrhových problémov.



Obr. Service Layers

Následky

Pre správny výber modelov služieb pre daný inventár je potrebné veľmi dobre poznať funkčné hranice inventára. To sa dá dosiahnúť opakovanými servisne orientovanými analýzami. Niekedy je dokonca potrebné upraviť už vytvorené služby. Celý tento proces môže viesť k predlženiu a sťaženiu definície inventára. Po vytvorení vrstiev je potrebné donich zaradiť všetky následne vytvorené služby, čo môže predstavovať problém, pretože je neskôr obtiažné meniť štruktúru vrstiev.

* + 1. Inventory Standardization Patterns

**Canonical protocol**

Problém

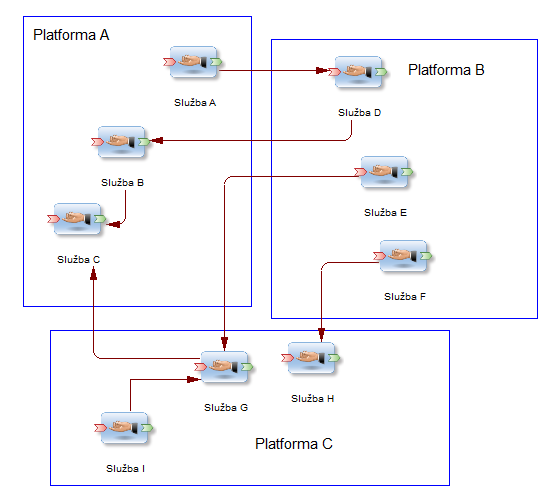
Pre vykonanie niektorých úloh je potrebné použitie viacerých služieb. Pri takomto vykonávaní musia služby medzi sebou komunikovať pomocou určitých komunikačných technológií. Ak ale služby používajú rôzne technológie, stávajú sa nekompatibilnými čo im znemožňuje komunikáciu. Takáto situácia sa dá riešiť pomocou programu, ktorý predstavuje rozhranie medzi dvomi technológiami, avšak takéto riešenie je nežiadúce.

Riešenie

Technologická architektúra, ktorá limituje možnosti použitia komunikačných technológií na jedinú je riešením takejto situácie. Služby v rámci inventára sú nútené použiť túto technológiu a tým pádom budú v každom prípade kompatibilné. Pri aplikácii tohoto vzoru je kľúčový výber komunikačnej technológie. Štandardnou platformou sú webové služby. Webové služby využívajú štandardné protokoly na výmenu správ a na transport správ (napr. HTTP a SOAP), ktoré sú bežne podporované a nezávislé na dodávateľovi. Napriek tomu, aj pri použití webových služieb, musí byť tento vzor aplikovaný, kvôli možnosti nezhody medzi jednotlivými verziami protokolov.

Následky

Pri výbere komunikačnej technológie treba brať do úvahy viacero faktorov. Je dôležité aby bol vybraný protokol stabilný, plne vyvinutý, aby jeho podpora pokračovala aj do budúcnosti a samozrejme aby bola jeho cena vyhovujúca pre podnik.



Obr. Canonical Protocol

**Canonical schema**

Problém

Pri komunikácii dvoch služieb musia obe služby poznať organizáciu a štruktúru dát. V jednom vývojárskom týme nieje problém dodržovať vopred dohodnutý formát, ale pri práci viacerých týmov môžu nastať problémy s rozdielnosťami použitých dátových modelov a tým spôsobenou nekompabilitou. Rozdielnosť dátových modelov sa dá vyriešiť pomocou transformácie, ale takýto prístup prináša zbytočné zaťaženie systému.

Riešenie

Zavedenie návrhových štandardov pre dátové modely kontraktov služieb v dostatočnej miere rieši tento problém. Vďaka tomu sú transformácie dátových modelov nepotrebné a záťaženie systému klesá. Tento vzor sa používa pri službách implementovaných ako webové služby, v podobe definície dátových modelov jazykom XML Schema expression language. V prípade, že je použitý tento jazyk, definujú XML schema definície rovnaké typy dokumentov alebo množín informácií pre dodržanie súladu komplexných a jednoduchých dátových typov v rôznych kontraktoch služieb. Po zavedení štandardných schém, je tento vzor realizovaný formálnym procesom v ktorom sú navrhované kontrakty služieb, čo zaručuje aplikáciu princípu štandardizovaného kontraktu služieb.

Následky

Vo väčších spoločnostiach je výhodnejšie štandardizovať dátové modely pre jednotlivé domény, vďaka čomu sa aj lahšie spravujú.

**Canonical Expression**

Problém

Konvencie použité pri zapisovaní funkcionality a kontextu danej služby v jej kontrakte sa môžu líšiť. Hlavna ak sú služby tvorené rôznymi týmami. V takom prípade sa môže stať, že nebudú kontrakty služieb správne pochopené a tým pádom ani správne použité.

Riešenie

Riešením je stanovenie menných a funkčných konvencií ako štandard pre vyjadrovanie kontextu a funkcionality služieb. Tento štandard by mali vývojári dodržiavať, čím sa zabezpečí správne pochopenie kontraktov služieb.

Následky

Aplikácia tohoto vzoru je jednoduchá a môže sa zdať, že nieje až tak dôležitá. Avšak vo veľkej miere znižuje riziko zbytočných chýb. Vďaka ľahkej aplikovatelnosti môže byť tento vzor nasadený ako štandard pre celú spoločnosť.

* 1. Vzory logických vrstiev inventára

Vrstvy služieb organizujú služby v rámci inventára do logických jednotiek. Každá vrstva je založená na určitom type služieb takže reprezentuje množinu služieb daného typu. Tieto typy korešpondujú s klasifikáciou podľa modelov služieb.

**Utility Abstraction**

Problém

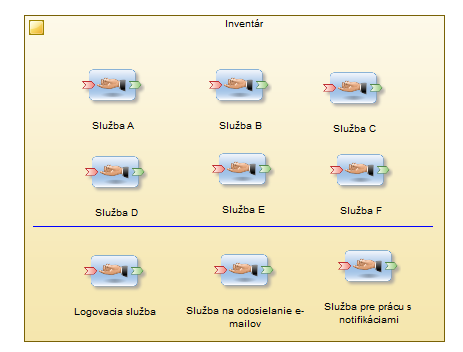
V každej službe sa používa tzv. pomocná logika, ktorá nemá nič spoločné s biznis logikou danej služby. Takáto logika sa používa na prácu s rôznymi technológiami, dátabázami a inými zdrojmi funkcionality. Tento typ logiky je určený k znovupoužitelnosti. Problém nastáva ak sa v jednej službe skombinujú biznis logika a pomocná logika, a to preto, lebo sa tým znemožňuje znovupoužitelnosť pomocnej logiky.

Riešenie

Riešním tejto situácie je vytvorenie agnostických služieb, ktoré vsebe nezahŕňajú žiadnu biznis logiku. Takéto služby obsahujú iba pomocnú logiku, pomocou ktorej vedia riešiť určitú všeobecnú úlohu, vďaka čomu sú znovupoužitelné. Služby tohoto typu sú zvyčajne tvorené technologickými odborníkmi a k ich tvorbe niesú potrebné znalosti odborníkov na biznis logiku.

Následky

Nasadenie tohto vzoru do inventára môže skomplikovať kompozíciu služieb. Ďalej môže znížiť výkonnosť systému, pretože sa zvýši počet služieb a aj prebiehajúca komunikácia medzi nimi. Avšak takéto riešenie môže byť bližšie vývojárom a aj celým orgnizáciám, pretože výhody modularity sú dobre známe z objektovo a aspektovo orientovaného programovania.



Obr. Utility Abstraction

**Entity Abstraction**

Problém

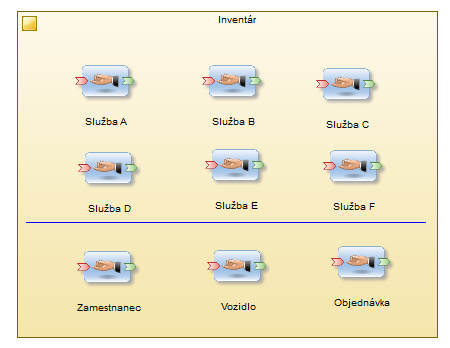
Pri vytváraní biznis logiky sa často stáva, že sa kombinuje logika ktorá by mohla byť znovupoužiteľná s logikou špecifickou pre daný proces. Takéto viazanie potencionálne znovupoužiteľnej logiky do služby, ktorá je vhodná iba na jeden účel vedie k opakovaniu toho istého kódu vo viacerých službách.

Riešenie

Každý podnik pracuje s rôznymi predmetmi záujmu ako napríklad produkty podniku, zamestnanci, zákazníci alebo iné organizácie. Tieto predmety, nazývané aj biznis entitami sa väčšinou s pribúdajúcimi úlohami alebo zmenou biznisu nemenia. Preto je výhodné stavať služby na základe týchto entít. Takéto služby sú znovupoužiteľné a nezávislé na ostatných službách zameraných na biznis procesy.

Následky

Tento vzor je najvhodnejšie aplikovať v čo najskorších fázach plánovania. Pretože neskôr by mohol spôsobiť zmeny v celej štruktúre podniku. Pri modelovaní a návrhu je potrebné vynaložiť maximálnu pozornosť.



Obr. Entity Abstraction

**Process Abstraction**

Problém

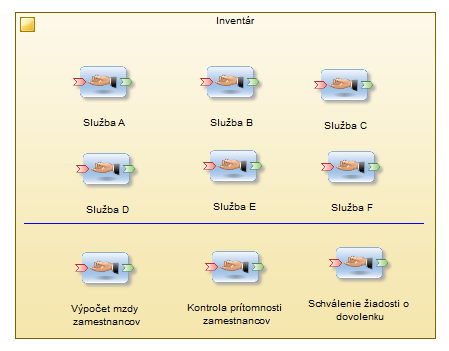
V prípade, že sú služby tvorené viacerými týmami, môže nastať situácia, že služby v sebe zahrňujú agnostickú a neagnostickú logiku. Toto vedie k zníženej možnosti znovupoužitelnosti agnostickej logiky a ťažšej správe služieb ak sa o biznis entity a biznis procesy starajú rôzny odborníci.

Riešenie

Vytvorenie vrstvy, ktorá je zodpovedná za workflow a kompozičnú logiku je riešením tohoto problému. Vďaka tejto vrstve je podnik schopnejší adaptácie na zmeny v biznise, pretože biznis procesy sú najčastejším predmetom zmien. Vrstva stanovená vzorom Enity Abstraction s agnostickou logikou tým pádom ostáva nezmenená.

Následky

Služby z tejto vrstvy spĺňajú svoje úlohy kompozíciou agnostický služieb z inej vrstvy. Zmeny v biznise sa reflektujú iba v zmene kompozicií týchto služieb. Organizácia, ktorá chce tento vzor použiť, musí byť pripravená na implementáciu kompozicií služieb aby mohol tento vzor fungovať.



Obr. Process Abstraction

* 1. Vzory Centralizácie inventára

Táto skupina vzorov sa zaoberá fyzickými aspektami architektúry orientovanej na služby. Na rozdiel od predchádzajúcich vzorov nieje nevyhnutná pre základný inventár služieb, ale je odporúčaná.

**Process Centralization**

Problém

Špecifická biznis logika by mala byť umiestnená v individuálnych službách, tak ako to je popísané vo vzore Process Abstraction. Avšak v podnikoch s väčším množstvom inventárov služieb sa často stáva, že tieto služby sú rozmiestnené vo viacerých inventároch, čiže majú rôzne fyzické umiestnenie. Problém nastáva ak treba vykonať niejaké zmeny v biznis procesoch, pretože sa tým, že sú služby uložené na rôznych miestach predĺžuje čas na vykonanie týchto zmien.

Riešenie

Uloženie biznis process logiky na jedno miesto, čiže jej centralizácia, je riešením tohoto problému. Na centralizáciu a jej správu je vhodné použiť na to určenú platformu, ktorá sa stará aj o hosting a vykonávanie danej logiky.

Následky

Nasadenie tohoto riešenia do podniku môže byť drahé, a problém môžu predstavovať aj hardvérové požiadavky potrebnej platformy. Preto je výhodné použitie tohoto vzoru pri plánovaní architektúry.

* 1. Vzory implementácie inventára

Vzory tejto skupiny riešia problémy na úrovni implementácie inventárov.

**Inventory Endpoint**

Problém

Často sa stáva, že je potrebné použiť niektoré služby mimo inventára. Čiže ich používajú služby z iných inventárov alebo služby iných spoločností. V takom prípade sa vynárajú otázky ohľadom bezpečnosti, privátnosti alebo interoperability.

Riešenie

Ako riešenie môže poslúžiť špeciálna služba postavená do role vstupného bodu inventára služieb. Takáto služba poskytuje prístup k službám inventára pre externých užívateľov a môže obsahovať logiku potrebnú pre umožnenie komunikácie medzi internými a externými službami. Kontrakty tejto špeciálnej služby a služieb ku ktorím poskytuje prístup môžu byť rovnaké, rozdiel môže spočívať iba v inak nastavených pravidlách alebo charakteristikách potrebných pri externej interakcii.

Následky

Tento vzor prináša väčšiu slobodu pri vývoji a riadení služieb, ale na druhej strane zvyšuje náklady na správu, pretože sa pri jeho aplikácii vytvárajú nové služby a ich kontrakty. Pri tvorbe viacerých vstupných bodov môže byť výhodné vytvorenie nového týmu iba pre tieto účely.

* 1. Vzory riadenia inventára

Vzory tejto skupiny návrhových vzorov riešia problémy pri návrhu inventára s ohľadom na zjednodušenie budúceho riadenia inventára.

**Metadata Centralization**

Problém

Pri tvorbe inventára služieb sa vždy môže stať, že funkcionalita vyvíjanej služby je už obsiahnutá v niejakej existujúcej službe. Čo vedie k problémom popísaných pri vzoroch Logic Centralization, Service Normalization a všeobecne k menej efektívnemu, redundantnému a denormalizovanému inventára.

Riešenie

Tento problém sa dá riešiť registrom služieb, ktorý je centrálnym prvkom inventára. V tomto registry sú pri nasadzovaní registrované všetky existujúce a práve vyvíjané služby. Pričom tieto registrované služby musia byť dostatočne popísané. Vývojáry po aplikovaní tohoto vzoru najprv vyhľadajú či neexistuje alebo nieje vo vývoji služba, ktorú potrebujú, a do jej vývoja sa pustia iba ak neexistuje ani v jednom stave. Aplikácia tohoto vzoru spočíva v použití princípu vyhľadatelnosti služieb pri návrhu kontraktu služieb. V štandardizovaní dokumentácie služieb a implementácii registra služieb.

Následky

Po aplikovaní tohoto vzoru je kľúčové udržiavanie aktuálnosti registra služieb a dodržovanie popísaných štandardov, pretože v opačnom prípade stráca register služieb zmysel. V niektorých prípadoch môže byť výhodné poverenie niektorého zamestnanca alebo celú skupinu udržiavaním aktuálnosti registra.

**Canonical Versioning**

Problém

Služby sa vo väčšine prípadov časom vyvíjajú, tým pádom prechádzajú vývojom aj kontrakty služieb. V takých prípadoch je na odlíšenie jednotlivých kontraktov výhodné označovanie verzií. Avšak rôzne vývojárske tými môžu použiť rôzne konvencie pre označovanie verzií a tým pádom môžu nastať problémy pri vývoji a znovupoužití služieb a celkového vývoja inventára.

Riešenie

Riešením problému môže byť stanovenie konvencií pre označovanie verzií služieb v danom inventáry. Vďaka tomu sa zaručí konzistentnosť riadenia služieb a ich vývoja a tiež štandardizáciu kontraktov. Aplikácia tohto vzoru spočíva vo výbere stratégie označovania verzií.

Následky

Jediným rizikom tohoto riešenia je, že ho nebudú tými dodržovať a budú sa vyhýbať jeho používaniu pomocou iných techník alebo ostanú pri vlastných konvenciách označovania. Úspešná aplikácia sa dá dosiahnuť vytvorením role v spoločnosti, ktorá bude mať za úlohu sledovať dodržiavanie systému verzií.

1. Aplikácia vzorov

Asdasd

* 1. Aplikácia vzoru Service Normalization

Aplikácia tohoto vzoru spočíva v preverení všetkých služieb daného inventára a odhaľovaní prekrývania ich kontextov. Pravidlá pre kontexty služieb je potrebné stanoviť už vo fáze analýzy a návrhu inventára, a následne sa musia dodržovať počas jeho implementácie. V praxi to vyzerá tak, že pri návrhu služieb sa vždy najprv skontroluje či daná funkcionalita nespadá do kontextu niektorej už existujúcej služby, a ak áno tak sa iba doplní do tejto služby.

Ako už bolo spomínané, pri veľkých inventároch s veľkým množstvom služieb prebieha tento proces iteratívne, pričom každá iterácia preveruje služby daného biznis procesu. Avšak pri inventároch, ktorích počet služieb dovoľuje postupné preverenie všetkých služieb, sa tento proces môže vykonať v jednom kroku. Takto to bolo aj pri aplikácii tohoto vzoru na inventár spoločnosti v praktickej časti tejto práce. Ako príklad sa môže uviesť návrh služieb pre entity systémov. Každá služba entity mala obmedzenú funkcionalitu iba na kontext danej entity, čiže žiadna služba entity „A“ nemohla v žiadnom prípade meniť stav inej entity „B“. Týmto sa dosiahlo dodržanie kontextov služieb a tým pádom aj normalizácia inventára služieb.

* 1. Aplikácia vzoru Logic Centralization

Vzor Logic Centralization je akýmsi pokračovaním predchádzajúceho vzoru Service Normalization. Ako bolo spomínané, Service Normalization zaručuje dodržovanie kontextov služieb vo fáze návrhu systému, Logic Centralization však vynucuje dodržovanie kontextov služieb pri pridávaní nových služieb do existujúceho inventára. Pri hotovom inventáry je pravdepodobné, že nová funkcionalita už je obsiahnutá v niektorej zo služieb, alebo, že nová funkcionalita patrí do kontextu už existujúcej služby. Ak daná funkcionalita už je implementovaná, tak ju treba v každom prípade použiť, kedže znovupoužiteľnosť je jedna z kľúčových vlastností architektúry orientovanej na služby. V druhom prípade, teda ak funkcionalita patrí do kontextu existujúcej služby, nieje potrebné vytvárať novú služby, stačí doplniť funkcionalitu existujúcej služby. Dodržanie tohoto vzoru sa dá dosiahnúť vydaním nariadenia pre programátorov služieb.

Po implementácii invetára v praktickej časti práce, boli zistené určité nedostatky vo funkcionalite systémov, a tento vzor bol aplikovateľný bez problémov a ťažkostí. Indikátorom toho je fakt, že po prvotnej implementácii neboli pridané žiadne nové služby. Pri službách entít boli často využívané agnostické služby pre operácie typu CRUD. Univerzálne služby boli tiež navrhované na znovupoužiteľnosť, ako napríklad služba na odosielanie e-mailov, ktorá bola použitá vo viacerých biznis procesoch. Nová funkcionalita patrila vždy do kontextu niektorej z už existujúcich služieb, takže nebolo potrebné vytvárať novú službu ani v jednom prípade. Tieto dôsledky neboli dosiahnuté iba vďaka tomuto vzoru, ale aj vzoru Service Normalization, podľa ktorého bol celý inventár navrhnutý.

* 1. Aplikácia vzoru Service Layers

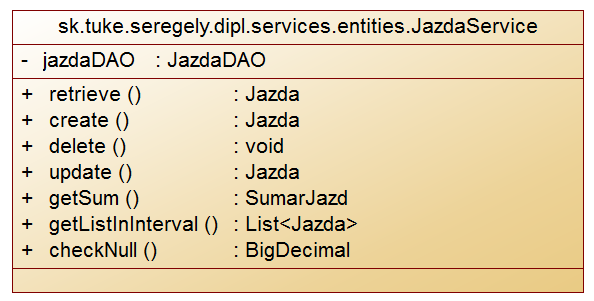
Ako to bolo popísané v teoretickej časti práce, je výhodné logicky rozdeliť inventár na vrstvy služieb s podobnou funkciou. Najčastejšie delenie služieb je na znovupoužiteľné a jednoúčelové. Ďalším krokom v takomto delení je rozdelenie na:

* služby entít (podľa vzoru Entity Abstraction)
* univerzálne služby (podľa vzoru Utility Abstraction)
* služby procesov (podľa vzoru Process Abstraction)

Pričom služby entít a univerzálne služby by mali byť znovupoužiteľné, a služby procesov by mali mať jeden účel. Aplikácia tohoto vzoru spočíva v logickom rozdelení jednotlivých vrstiev, čo vedie ku zvýšenej konzistencii služieb patriacich do jednej vrstvy. Takéto delenie sa dá uviesť aj do fyzickej roviny, čo sa dá preukázať delením služieb v jednotlivých informačných systémoch praktickej časti práce. Napríklad v aplikácii GpsMonitoring sú implementácie služieb rozdelené do balíkov podľa ich funkcie, čo na platforme Java znamená aj fyzické delenie v súborovom systéme. Implementácie služieb entít sú umiestnené v balíku sk.tuke.seregely.dipl.services.entities a univerzálne služby sú v balíku sk.tuke.seregely.dipl.serveice.utilities. Služby procesov sa v tejto aplikácii nenachádzajú, čo vyplýva z povahy ich implementácie, ale to bude bližšie popísané pri aplikácii vzoru Process Abstraction. Pri tomto vzore však nieje dôležité fyzické oddelenie služieb patriacich do iných vrstiev, takže iné platformy niesu v nevýhode pri aplikácii tohoto vzoru.

* 1. Aplikácia vzoru Entity Abstraction

Tento vzor predstavuje jednu z vrstiev definovaných vo vzore Service Layers. Služby tejto vrstvy by mali byť znovupoužiteľné a všetky by mali byť odvodené od entít systému. Pri aplikácii tohoto vzoru je potrebné identifikovať takéto služby a logicky ich oddeliť od ostatných služieb inventára. Oddelenie sa dá dosiahnuť jednoducho umiestnením implementácií služieb do jedného balíka, alebo zavedením mennej konvencie pre takéto služby. V praktickej časti práce boli služby oddelené oboma spôsobmi. Zaviedla sa menná konvencia, ktorá sa skladala vždy z názvu entity, pre ktorú bola služba vytvorená a postfixu „Service“. Ďalej je takáto služba umiestnená do balíku vyčleneného pre túto vrstvu služieb. Na nasledujúcom obrázku je takáto služba v UML notácii.



Ako vidno na predchádzajúcom obrázku X.Y, služba má okrem iných všetky z CRUD operácií. Takto to je pri všetkých službách entít, pretože sú tieto operácie nevyhnutné pre prácu s entitami . Pri identifikácii entít sa dá vychádzať aj z dátového modelu ak už je navrhnutý, pretože sa väčšinou tabuľky v databáze rovnajú entitám systému. Na nasledovnom obrázku sú zvýraznené všetky služby entít v inventáry služieb.

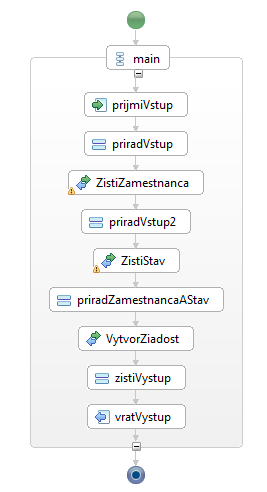
* 1. Aplikácia vzoru Utility Abstraction

Podobne ako predchádzajúci vzor, predstavuje vzor Utility Abstraction realizáciu ďalšej vrstvy služieb, podľa vzoru Service Layers. Táto vrstva obsahuje služby, ktoré sa nijak neviažu na biznis procesy a ani na entity systémov. Takéto služby sú znovupoužiteľné, a podľa miery nezávislosti na procesoch a entitách systémov, oddelené od ostatných služieb. V ukážkovom inventáry, implementovanom v praktickej časti práce sme identifikovali nasledovné služby nezávislé na procesoch a entitách firmy:

* EmailSenderService – Služba zodpovedná za odosielanie e-mailov. Táto služba sa využíva napríklad pri informovaní zamestnancov o neschválenej dovolenke v procese schvaľovania dovoleniek, alebo informuje o výhradách kvôli ktorím neboli schválené jazdy vodiča motorového vozidla v procese kontroly jázd daného vodiča.
* GpsExportService a AttendanceExportService – Tieto dve služby sú zodpovedné za export schválených jázd a export pracovného vyťaženia zamestnancov do systému SAP spoločnosti. Ich úlohou je zapisovanie do tabuliek rozhrania systému SAP, ktorý ďalej tieto údaje spracúva.

Prvá univerzálna služba EmailSenderService je úplne nezávislá na procesoch a entitách systému, preto je umiestnená v samostatnej webovej aplikácii s názvom Utilities, ktorý je vyhradený iba na univerzálne služby. Táto služba môže byť použitá hocikým v spoločnosti, niesú naňu kladené žiadne obmedzenia. Ďaľšie služby patria tiež medzi univerzálne služby, ale kvôli ich viazaniu na entity informačných systémov boli umiestnené v príslušných webových aplikáciach. Oddelené od ostatných služieb boli iba umiestnením v balíkoch vyhradených pre univerzálne služby. Na nasledovnom obrázku X.Y je zobrazená oddelená vrstva univerzálnych služieb.

* 1. Aplikácia vzoru Process Abstraction

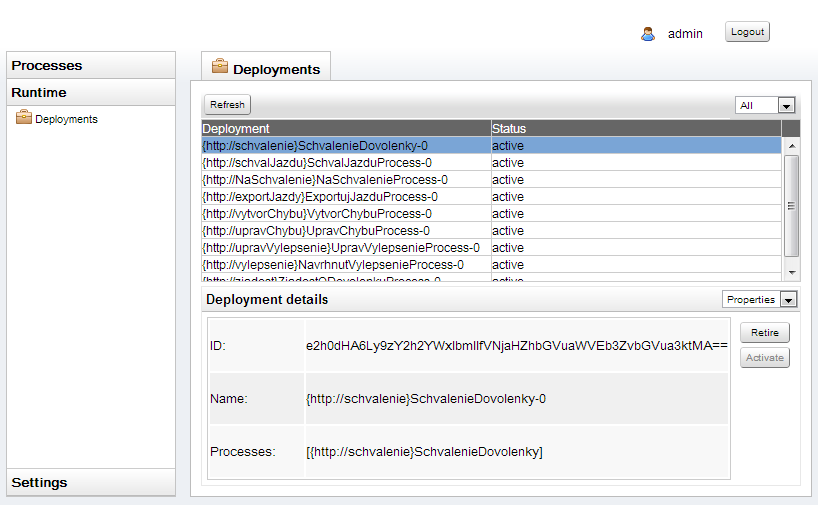
Služby s biznis logikou reprezentujúcou biznis procesy danej firmy, sú umiestnené v poslednej vrstve služieb, ktorá je vytvorená nasadením tohoto vzoru. Služby tejto vrstvy vo väčšine prípadov predstavujú sekvenciu volaní služieb iných vrstiev, ale obsahujú aj vlastnú logiku. Takéto služby väčšinou niesú agnostické, kedže predstavujú určitý biznis process, ale niektoré časti biznis procesov sa môžu opakovať a v tom prípade je vhodné vytvoriť podproces ktorý sa jednoducho zavolá v každom procese v ktorom je obsiahnutý. Služby tejto vrstvy môžu byť implementované ako klasické webové služby iných vrstiev, ale výhodnejšie, a častokrát aj jednoduchšie je použitie niektorého z jazykov na modelovanie biznis procesov. Najpoužívanejšími možnosťami sú jazyk BPMN vo verzii 2.0 a BPEL podľa špecifikácie WS-BPEL. BPMN je vhodný na modelovanie a spúšťanie biznis procesov v ktorích sú ako prvky logiky použité aj iné systémy, nielen webové služby, medzi možnosti patria aj činnosti respektíve rozhodnutia vykonané človekom. Ďalej je výhodný pre dlho trvajúce biznis procesy, a pre komplexnejšie riešenia zachytávania chýb. Tento jazyk je čitateľnejší pre menej technicky zdatných ľudí. Na druhej strane, jazyk BPEL je určený na orchestráciu webových služieb. Má viacero možností pre prácu s webovými službami, ako napríklad asynchrónne volania webových služieb, ale je ťažšie čitateľný pre technicky neznalých pracovníkov. Kedže príkladový inventár je tvorený webovými službami, zvolený bol práve jazyk BPEL. Zložitejšie biznis procesy boli implementované pomocou BPEL procesov, ktoré sú založené na Xml súboroch a spúšťané pomocou BPEL. Výhodou takéhoto riešenia je, že postupom času sa takto implementované biznis procesy dajú ľahko meniť, kedže je možné pridávanie alebo mazanie jednotlivých prvkov procesu, poprípade zmena ich poradia. Na vývoj procesov bol použitý nástroj BPEL Designer, nainštalovaný ako doplnok vývojového prostredia Eclipse. Tento doplnok uľahčuje implementáciu vďaka jeho grafickému rozhraniu a vizualizácii procesov. K tvorbe procesov však nieje nevyhnutný.

Na obrázku X.Y je príkad jednoduchého biznis procesu skladajúceho sa z volaní troch webových služieb. Konkrétne ide o process podania žiadosti na dovolenku v informačnom systéme Attendance Tracking. V prvom kroku po prijatí vstupu sa nastavia vstupné premenné prvej služby na kontrolu reálnosti zadaného zamestnanca. V ďalšom kroku sa vyberie požadovaný stav dovolenky, v tomto prípade to je stav „podana ziadost“. Posledné volanie služby s názvom „VytvorZiadost“ vytvorí inštanciu dovolenky s nastavenými atribútmi získanými z predošlých volaní služieb a vstupu používateľa.

Takto implementované biznis procesy sú jednoznačne oddelené od ostatných vrstiev inventára a takto tvoria samostatnú vrstvu, čím spĺňajú požiadavky vzoru Process Abstraction.

* 1. Aplikácia vzoru Process Centralization

Tento vzor je úzko spätý so vzorom Process Abstraction, pretože rozhodnutie vykonané v spomínanom vzore určuje platformu na ktorej budú služby procesov hostované a vykonávané. Pri BPMN procesoch sú najpoužívanejšie procesné enginy Activiti a jBPM. Obe sú open-source riešenia a bežne sa používajú aj v produkčnom prostredí. Pri nami zvolenej platforme BPEL sú vyhovujúcimi riešeniami open-source projekt Apache ODE a riešenie od Oraclu, BPEL process Manager. Kvôli viazanosti na Oracle produkty sme sa radšej priklonili k riešeniu od Apache Foundation. Aplikačný server bol zvolený už skôr, a to konkrétne JBoss. Pre inštaláciu BPEL prostredia je potrebné vytvoriť databázu, nakonfigurovať spojenie naňu a nainštalovať samotné prostredie na aplikačný server. Tento postup nám však uľahčila komunita okolo JBoss, pretože vďaka nej existuje riešenie postavené na Apache ODE, ktoré je nakonfigurované na JBoss server s názvom Riftsaw. Pomocou tejto aplikácie je možné prezerať nainštalované a spustené procesy, nahrávať nové a vymazávať existujúce procesy. Čiže použitím tohoto prostredia aplikujeme vzor Process Centralization v našom inventáry. Avšak treba podotknúť, že procesy implementované podľa vzoru Process Abstraction sú plne kompatilbilné štandardu WS-BPEL, takže niesú viazané na žiadné konkrétne BPEL prostredie. Na nasledujúcom obrázku je zobrazené rozhranie tohoto procesného enginu so zobrazenými aktívnymi procesmi.



* 1. Aplikácia vzoru Metadata Centralization

Cieľom vzoru Metadata Centralization je vytvorenie centrálneho bodu invenátaru, ktorý umožňuje prezeranie kontraktov existujúcich služieb a publikovanie nových služieb. Takýto centrálny bod sa realizuje pomocou registra služieb. Štandardom v tomto smere je špecifikácia UDDI aj keď jej reálne využitie je diskutabilné. Ďalšou možnosťou je štandard ebXML, pričom obe technológie sú založené na XML. Väčšina moderných aplikačných servrov však má implementovaný mechanizmus na tieto účely, platí to napríklad pri servroch Glassfish, Weblogic a Websphere. Po zhodnotení kladov a záporov, s prihliadaním na fakt, že príkladový inventár je určený na použitie v intranete spoločnosti, bol vybraný základný mechanizmus pre publikovanie a prehliadanie kontraktov služieb, poskytovaný aplikačným servrom JBoss. Táto technológia je implementovaná v aplikačnom servry, takže nieje potrebná ďalšia konfigurácia. Tento mechanizmus registrácie služieb je však použitý iba pri dvoch vrstvách služieb podľa vzoru Service Layers. Ako to bolo spomenuté pri aplikácii vzoru Process Centralization, vrstva s procesnými službami je realizovaná pomocou BPEL enginu Apache ODE, a tento využíva na publikovanie a prehliadanie štandard UDDI.

Pre publikáciu webovej služby týmto sôsobom je potrebné triedu s implementáciou služby registrovať v súbore web.xml. Napríklad nasledovným spôsobom:

<servlet>

<display-name>**EmailSender**</display-name>

<servlet-name>**EmailSender**</servlet-name>

<servlet-class>**sk.tuke.seregely.dipl.util.services.EmailSender**</servlet-class>

</servlet>

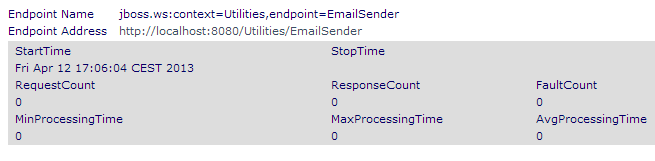
<servlet-mapping>

<servlet-name>**EmailSender**</servlet-name>

<url-pattern>**/EmailSender**</url-pattern>

</servlet-mapping>

Takto registrovaná webová služba je následne zobrazená medzi aktívnymi službami v rozhraní webových služieb aplikačného servra JBoss, ako na obrázku X.Y. Z obrázku je jasné, že služba patrí do projektu Utilities, že je implementovaná v triede EmailSender, ďalej je URL adresa jej kontraktu, a nakoniec sú v sivom rámčeku informácie o jej spustení a ďaľšie informácie o počte jej volaní a chybách.



* 1. Aplikácia vzoru Inventory Endpoint

Zmyslom vzoru Inventory Endpoint je zabezpečenie rozhrania inventára pre komunikáciu s externými systémami. Takéto systémy môžu byť v rámci firemného intranetu alebo aj vo vzdialených sieťach iných spoločností. Pri komunikácii v rámci spoločnosti je situácia jednoduchšia, pretože sú požiadavky na bezpečnosť komunikácie nižšie. Avšak pri komunikácii cez internet nestačí na plné zabezpečenie ani protokol https. V takom prípade je potrebné zváženie aplikácie vzorov zo skupiny Service Security Patterns [odkaz]. Služby navrhnuté podľa tohoto vzoru patria do vrstvy služieb vzoru Process Abstraction. V praktickej časti práce boli implementované nasledovné služby ako rozhrania inventára:

* JazdaServiceEndpoint – Služba pre externý systém moniturujúci aktuálnu pozíciu vozidiel. Po začatí jazdy vozidla tento systém inicializuje Jazdu.
* SuradnicaServiceEndpoint – Služba pre monitorovací systém vozidiel, v tomto prípade zaznamenáva aktuálnu pozíciu vozidla.
* DochadzkaServiceEndpoint – Služba zaznamenávajúca údaje z terminálu, na ktorom sa zamestnanci prihlasujú po príchode do zamestnania.
* GpsExport – Služba reprezentujúca rozhranie so systémom SAP, do ktorého sa zapisujú jazdy schválené v procese schvaľovania jázd.
* Ďaľšie služby na komunikáciu so systémom SAP zo systému Attendance Tracking – PnExport, PdExport, DovolenkaExport
  1. Aplikácia vzoru Canonical Protocol

Štandardnou formou realizácie aplikačnej logiky v architektúrach orientovaných na služby sú webové služby. Každá z plne vyvynutých platforiem má svoju realizáciu tohoto konceptu. Príkladový inventár bol navrhnutý do prostredia Java EE, a vtejto platforme je štandard pre webové služby JAX-WS. Existuje viacero implementácií tohoto štandardu, pre náš inventár bol vybraný open source rámec Apace CXF. Tento výber padol preto lebo táto implementácia ponúka kompletné riešenie pre webové služby a tiež preto lebo je podporovaná použitým aplikačným servrom JBoss. Po výbere prostredia pre služby je jednoduchšia aplikácia vzoru Canonical Protocol, pretože práve prostredie limituje možnosti komunikácie služieb. Pri webových službách je štandardom SOAP prenášaný sieťovým protokolom HTTP, respektíve HTTPS. Aplikácia tohoto vzoru spočíva v štandardizovaní jedného protokolu, za účelom zabezpečenia kompatibility služieb. Kvôli predídeniu neskorších nedorozumení je najlepšie neštandardizovať iba protokol, ale aj jeho verziu. Pri protokole SOAP je aktuálna verzia 1.2.

* 1. Aplikácia vzoru Canonical Schema

Pre udržanie kompatibility medzi webovými službami nestačí iba štandardizovať komunikačný protokol, je potrebné štandardizovať aj štruktúru dát ktoré si vymieňajú. Pri jednoduchých dátových typoch je situácia jednoduchšia pretože je ľahšie dodržať, že niektorá premenná je typu int v každej službe. Avšak pri komplexných dátových typoch sa môžu vyskytnúť rôzne reprezentácie tej istej entity. Ako príklad môžme uviesť entitu sumáru jázd vozidla:

<xs:complexType name=**"sumarJazd"**>

<xs:sequence>

<xs:element minOccurs=**"0"** name=**"obdobieOd"** type=**"xs:dateTime"** />

<xs:element minOccurs=**"0"** name=**"obdovieDo"** type=**"xs:dateTime"** />

<xs:element minOccurs=**"0"** name=**"spotreba"** type=**"xs:decimal"** />

<xs:element maxOccurs=**"unbounded"** minOccurs=**"0"** name=**"vodici"** nillable=**"true"** type=**"tns:vodic"** />

<xs:element miOccurs=**"0"** name=**"vzdialenost"** type=**"xs:decimal"** />

<xs:element maxOccurs=**"unbounded"** minOccurs=**"0"** name=**"zakazky"** nillable=**"true"** type=**"tns:zakazka"** />

</xs:sequence>

</xs:complexType>

Pri podobnom komplexnom type môže nastať nedorozumenie veľmi ľahko, napríklad použitím iného dátového typu pre element „obdobieOd“ pretože existuje viacero typov pre dátum, alebo jednoducho vynechaním niektorého atribútu. Aplikácia vzoru Canonical Schema zamedzuje takýmto nedorozumeniam. Pre jej realizáciu je potrebné jednoznačne definovať dátové typy entít, a následne pri vývoji webových služieb tieto typy vždy používať.

* 1. Aplikácia vzoru Canonical Expression

Menné konvencie vedia vo veľkej miere uľahčiť pochopenie kontextu a funkciu služby. Vzor Canonical Expression má za úlohu štandardizovať názvy služieb a ich metód. Najviac sa táto štandardizácia prejavuje pri službách patriacich do vrstvy entít. Ako už bolo spomenuté tieto služby sú pomenované podľa entity ktorú reprezentujú a ich metódy sú minimalne CRUD operácie:

* Create – pre vytvorenie novej inštancie
* Retrieve – pre výber už existujúcej inštancie
* Update – pre zmenu inštancie
* Delete – pre vymazanie

Pri službách ostatných vrstiev inventára sa takáto univerzálna konvencia nedá zaviesť pretože ich metódy sú špecifické a väčšinou sa neopakujú. Avšak vždy sa dá štandardizovať aspoň konvencia tvorby názvov. V príkladovom inventáry bola použitá konvencia známa z platformy Java, pričom prvé slovo označuje činnosť danej metódy a nasledovné označujú návratovú hodnotu alebo bližšie špecifikujú funkcionalitu. Podľa tejto konvencie sa napríklad metóda vracajúca zoznam jázd označí ako getRidesList().

1. Záver

Záver by mal zachytiť jasnú a presnú prezentáciu dedukcií vychádzajúcich z jadra práce. Musí byť vecnou sumarizáciou vlastného prínosu alebo pohľadu na riešenú problematiku. Zahrnúť možno aj kvantitatívne údaje, ale podrobnosti by sa nemali uvádzať. Záver nemá obsahovať nič, čo nie je v texte práce a musí nadväzovať na úvahy a argumenty v texte práce.

V závere je vhodné poukázať na ďalšie otvorené (doteraz nevyriešené) problémy, ktorým je vhodné venovať pozornosť a ktoré presahujú odporúčaný rozsah diplomovej práce. Odporúčané sú popisy ďalších navrhovaných aktivít, ktoré priamo vyplývajú zo záverov alebo skúseností získaných v priebehu spracovania práce.

Zoznam použitej literatúry

Všetky dokumenty, ktoré v práci použijete, je potrebné zoradiť do zoznamu pozostávajúceho z  bibliografických odkazov (b. o., en bibliographic reference), ktorý označujeme napr. **Zoznam použitej literatúry**. Pre tvorbu zoznamov použitej literatúry platia štandardy. Cieľom je, aby zo zoznamu použitej literatúry bolo možné jasne identifikovať použitý zdroj a aby ho bolo možné bez ťažkostí opäť vyhľadať.

Hlavným zdrojom údajov pre tvorbu bibl. odkazov je **titulný list** (tzn. prvý list v knihe, kde sú uvedené údaje o názve autorovi atď.), príp. jeho rub. Odkazy sa môžu týkať knižných, časopiseckých a iných zdrojov informácií (zborníky z konferencií, patentové dokumenty, normy, odporúčania, kvalifikačné práce, osobná korešpondencia a rukopisy, odkazy cez sprostredkujúci zdroj, elektronické publikácie), ktoré boli v práci použité.

**Technika citovania** určuje spôsob, akým označujeme citácie v dokumente, pričom podľa normy (pozri STN ISO 690 ) existuje viacero spôsobov citovania:

* metóda číselných citácií (citácie umiestňujeme v práci podľa odkazových čísel (číslo zo Zoznamu použitej literatúry), ktoré zodpovedajú poradiu citácií v texte),
* citácie v poznámkach,
* metóda prvého údaja a dátumu. (citácie umiestňujeme v práci abecedne podľa prvého údaja (meno autora + rok vydania)).

Pri metóde **číselných citácií** sa v zozname bibliografických odkazov každé citované dielo uvádza v tom poradí, v akom bolo uvedené a číslované v texte. Číslované odkazy v texte sú uvedené v zátvorkách a odkazujú na dokumenty v takom poradí, v akom sa citujú po prvýkrát. Nasledujúce citácie dostávajú také isté číslo, ako má prvá citícia. Ak sa citujú osobitné časti dokumentu, môžu sa za číslom citácie uviesť čísla strán. Metóda číselných citácií je podrobne popísaná v norme STN ISO 690 (pozri ).

**Príklad zoznamu použitej literatúry je uvedený na nasledujúcej strane.**

1. MIHALÍK, Ján – ZAVACKÝ, Jozef – GLADIŠOVÁ, Iveta: Signály a sústavy : Návody na cvičenia. Košice : TU-FEI, 2004. 241 s. ISBN 80-8073-138-1
2. CIMBALA, Roman - BALOGH, Jozef - DŽMURA, Jaroslav: Diagnostika výkonových transformátorov s využitím prvkov umelej inteligencie 1. In: Elektrotechnický magazín ETM. roč. 14, č. 1 (2004), s. 8-9.
3. KOVAĽAKOVÁ, Mária - NOVÁK, Ladislav - STANČÁKOVÁ, Anna: Vplyv prímesi chrómu na proces hydrogenácie a dehydrogenácie FeB amorfných zliatin. In: 13. konferencia slovenských fyzikov : Zborník príspevkov. Košice : Slovenská fyzikálna spoločnosť, 2004. s. 145-146.
4. Therion Biologics Corporation, Cambridge, MA: Recombinant fowlpox virus and recombination vector. Inventors: Cohen L. K., Panicali; D. L. Int. Cl.5 C12N/701 United States Patent, 5093258. 03.03. 92.
5. ISO 690-2: 1997, Information and documentation – Bibliographic references - Part 2: Electronic documents or parts thereof.
6. STN ISO 690:1998 : Dokumentácia - Bibliografické odkazy - Obsah, forma a štruktúra.
7. Zákon č. 183/2000 Z.z. o knižniciach, o doplnení zákona Slovenskej národnej rady č. 27/1987 Zb. o štátnej pamiatkovej starostlivosti a o zmene a doplnení zákona č. 68/1997 Z.z. o Matici slovenskej.
8. Vyhláška č. 131/1997 Zb. Ministerstva školstva Slovenskej republiky zo 7. mája 1997 o doktorandskom štúdiu.
9. LAGOZE, C. a kol. The Open Archives Initiative Protocol for Metadata Harvesting [online]. Protocol Version 2.0 of 2002-06-14. Document Version 2004/10/12T15:31:00Z 2004 [cit. 2004-11-10]. Dostupné na internete: <http://www.openarchives.org/OAI/openarchivesprotocol.html>.
10. Elektronické diplomové a dizertačné práce SR: ETD SK. [online]. Košice : ETD SK, 2004. Aktualizované 14-2-2005 [cit 2005-03-10]. Dostupné na internete: <http://www.etd.sk/>.
11. UNESCO. The Guide to Electronic Theses & Dissertations [online]. Paris : UNESCO, c2001 [cit 2004-11-10]. Dostupné na internete: <http://etdguide.org/>.
12. HOGGAN, Daniele. 2002. Challenges, Strategies, and Tools for Research Scientists. In Electronic Journal of Academic and Special Librarianship [online]. 2002, vol. 3, no. 3 [cit. 2003-01-10]. Dostupné na internete: <http://southernlibrarianship.icaap.org/content/v03n03/Hoggan\_d01.htm>. ISSN 1525-321X
13. KOMOROVÁ, K. Výstava vzácnych kódexov. In Knižnica [online]. Martin : SNK, 2002 [cit. 2003-02-14], 2002, roč. 3, č. 2, s. 84. Dostupné na internete: <http://www.snk.sk/kniznica/kniznica.html>. ISSN 1212-5075
14. PARKER, Elliott. Re: Citing Electronic Journals. In : PACS-L (Public Access Computer Systems Forum) [online]. Houston (Tex.) : University of Houston Libraries, 24 November 1989; 13:29:35 CST [citované 2003-01-05]. Dostupné na internete: <telnet://brsuser@a.cni.org>.
15. BURAN, Daniel. 2003. Environmentálne informačné zdroje a služby v strednej a východnej Európe [elektronická pošta]. Správa pre: Mária MALÁ. 2002-11-15 [cit. 2003-01-05]. Osobná komunikácia.
16. GONDA, Vladimír: Ako napísať a úspešne obhájiť diplomovú prácu. Bratislava : Elita, 2003. 124 s. : il. ISBN 80-8044-076-X
17. Katuščák, Dušan : Ako písať záverečné a kvalifikačné práce. Nitra: Enigma, 2004. 162 s. il. ISBN 80-89132-10-3

Prílohy

1. CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe.
2. Používateľská príručka
3. Systémová príručka

Táto časť diplomovej práce je povinná a obsahuje zoznam všetkých príloh vrátané elektronických nosičov. Názvy príloh v zozname musia byt’ zhodné s názvami uvedenými na príslušných prílohách. Tlačené prílohy majú na prvej strane identifikačné údaje – informácie zhodné s titulnou stranou diplomovej práce doplnené o názov príslušnej prílohy (Systémová príručka, Používateľská príručka). Identifikačné údaje sú aj na priložených diskoch alebo disketách. Ak je médií viac, sú označené aj číselne v tvare I/N, kde I je poradové číslo a N je celkový počet daných médií.

Každá príloha začína na novej strane a je označená samostatným písmenom (Príloha A, Príloha B, ...). Číslovanie strán príloh nadväzuje na číslovanie strán v hlavnom texte.

Curriculum vitae

Táto časť je nepovinná. Autor tu môže uviesť svoje biografické údaje, údaje o záujmoch, účasti na projektoch, účasti na súťažiach, získané ocenenia, zahraničné pobyty na praxi, domácu prax, publikácie a pod.