MASARYKOVA UNIVERZITA

FAKULTA INFORMATIKY

DIPLOMOVÁ PRÁCE

NÁVRHOVÉ VZORY V SOA

Brno, 2011 Roman Laštovička

Prohlášení: Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Shrnutí: Cílem práce je podat ucelený náhled na návrhové vzory v servisně orientovaných architekturách. Součástí práce bude i praktická implementace vzorů v ukázkové aplikaci namodelované pomocí UML 2.0.

Klíčová slova: Servisně orientovaná architektura, návrhové vzory, UML, služba, inventář.

Obsah

1 Úvod

Bohužel jsem stále nebyl schopen věnovat diplomové práci dostatečné úsilí a tak odevzdávám pouze velmi malou část nehotové práce, protože je to jediná možnost, jak se vyhnout vyloučení ze studia a nepřijít tak o možnost využití opravného termínu obhajoby. Tímto se omlouvám všem hodnotícím za čas, o který jsem je připravil při posuzování této práce, od níž samozřejmě neočekávám jiné hodnocení než nevyhovující.

Práce obsahuje návrhové vzory pro tvorbu inventáře služeb v SOA a několik jejich předběžných diagramů v jazyce UML, které budou později zahrnuty v příloze.

2 Základní vzory pro tvorbu inventáře služeb

Enterprise Inventory (Podnikový inventář)

Jak mohou být služby dodávány, aby byla maximalizována schopnost jejich vzájemné spolupráce?

Problém: Dodáváním služeb od několika nezávislých vývojových týmů roste riziko produkce nekonzistentních služeb, které mají sníženou schopnost vzájemné spolupráce.

Řešení: Služby mohou být vytvářeny ve standardizované formě a mohou využívat architekturu společného inventáře pro celý podnik. V tomto inventáři mohou být volně a opakovaně využívány.

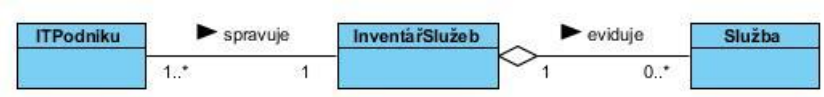
Aplikace: Podnikový inventář je vhodné namodelovat v předstihu a celopodnikové standardy je nutné aplikovat na služby dodávané různými vývojovými týmy.

Dopady: K definování podnikového inventáře služeb je nutná podrobná předběžná analýza. Následné požadavky na řízení mohou mít mnoho organizačních dopadů.

Principy: STANDARDIZED CONTRACT, ABSTRACTION, COMPOSABILITY.

Architektury: ENTERPRISE, INVENTORY.

UML diagram:



Domain Inventory (Doménový inventář)

Jak mohou být služby dodávány, aby schopnost jejich spolupráce byla maximální, když není možné docílit celopodnikové standardizace?

Problém: Vytvoření celopodnikového inventáře služeb může být pro některé podniky příliš náročné a pokusy o jeho navržení mohou výrazně ohrozit úspěšný přechod k SOA.

Řešení: Služby mohou být seskupeny do lépe zvládnutelných, oblastních servisních inventářů. Každý z inventářů může být standardizován a spravován nezávisle na ostatních.

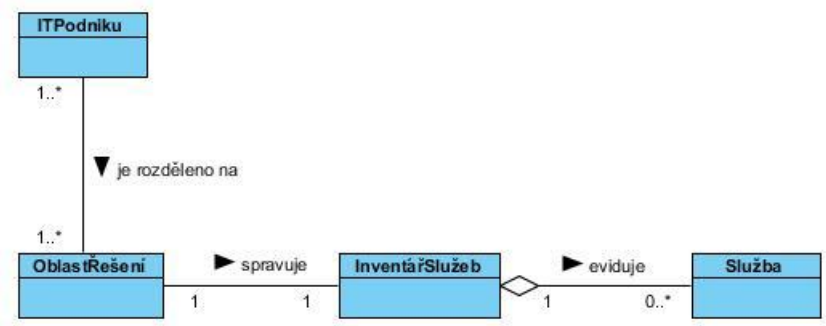
Aplikace: Musí být pečlivě stanoveny hranice mezi doménovými inventáři, podle nichž proběhne postupná přeměna.

Dopady: Rozdíly ve standardizaci mezi jednotlivými doménovými inventáři zvyšují požadavky na transformaci a potenciálně snižují celkový přínos přechodu k SOA.

Principy: STANDARDIZED CONTRACT, ABSTRACTION, COMPOSABILITY.

Architektury: ENTERPRISE, INVENTORY.

UML diagram:



Service Normalization (Normalizace služeb)

Jak můžeme navrhnout servisní inventář, abychom se vyhnuli redundanci služeb?

Problém: Při doplňování služeb do inventáře vzniká riziko vytvoření služeb s překrývajícími se funkčními hranicemi, které snižuje znovupoužitelnost služeb.

Řešení: Inventář služeb musí být navrhován s důrazem na sladění hranic jednotlivých služeb.

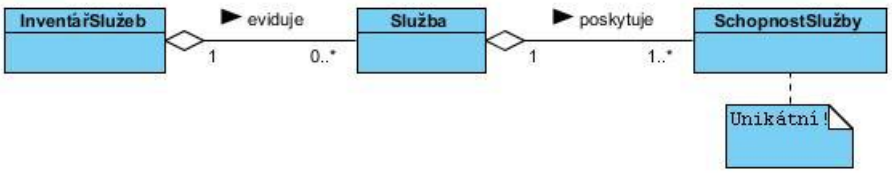
Aplikace: Hranice služeb jsou modelovány jako část procesu formální analýzy a řídíme se jimi při návrhu a správě servisního inventáře.

Dopady: Pro vytvoření nepřekrývajících se služeb musíme vynaložit dodatečné úsilí podpořené podrobnou předběžnou analýzou.

Principy: AUTONOMY.

Architektury: INVENTORY, SERVICE.

UML diagram:



Logic Centralization (Centralizace logiky)

Jak se můžeme vyhnout nesprávnému použití redundantní servisní logiky?

Problém: Pokud nejsou agnostické služby používány správným způsobem, může dojít k vytvoření redundantní funkcionality v dalších službách. Z toho plynou problémy s normalizací, vlastnictvím a správou služeb.

Řešení: Přístup ke znovupoužitelné funkcionalitě je omezen pouze na agnostické služby, které tuto funkcionalitu poskytují.

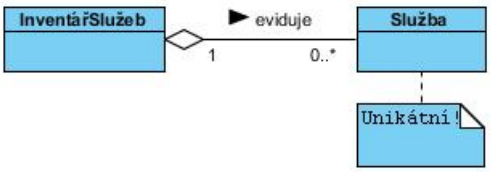
Aplikace: Agnostické služby musí být správně navrženy a spravovány a jejich použití musí být řízeno podnikovými standardy.

Dopady: Organizační záležitosti týkající se dřívějších projektů mohou vytvořit překážky při aplikaci tohoto vzoru.

Principy: REUSABILITY, COMPOSABILITY.

Architektury: INVENTORY, COMPOSITION, SERVICE.

UML diagram:



Service Layers
(Vrstvy služeb)

Jak můžeme služby v servisním inventáři organizovat na základě funkční podobnosti?

Problém: Libovolné definování služeb, které jsou dodávány a spravovány různými vývojovými týmy, může vést k nekonzistenci návrhu a neúmyslné funkční redundanci uvnitř servisního inventáře.

Řešení: Inventář služeb je strukturován do dvou nebo více logických vrstev, z nichž každá je zodpovědná za abstrakci logiky založené na společném funkčním typu služeb.

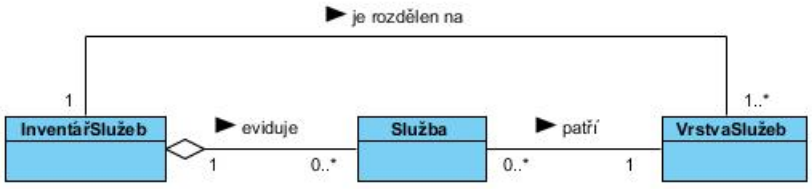
Aplikace: Na základě vybraných modelů služeb jsou vytvořeny vrstvy těchto služeb a modelovací a návrhové standardy pro jejich tvorbu.

Dopady: Je třeba akceptovat zvýšené úsilí, které je nutné vynaložit při předběžné analýze a při vytváření návrhových standardů.

Principy: REUSABILITY, COMPOSABILITY.

Architektury: INVENTORY, SERVICE.

UML diagram:



Canonical Protocol (Kanonický protokol)

Jak můžeme služby navrhnout, abychom se vyhnuli přepínání mezi protokoly?

Problém: Služby, které podporují různé komunikační technologie, ohrožují vzájemnou spolupráci, limitují počet potenciálních konzumentů a zvyšují nutnost přepínání mezi komunikačními protokoly.

Řešení: Ustanovíme jedinou komunikační technologii jako výhradní a primární médium pro komunikaci mezi službami.

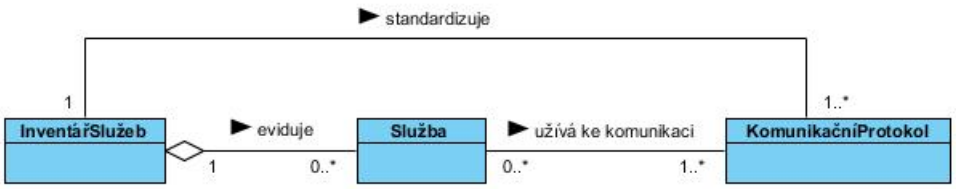
Aplikace: Použité komunikační protokoly včetně jejich verzí jsou standardizovány pro všechny služby uvnitř servisního inventáře.

Dopady: Architektura inventáře se standardizovanými komunikačními protokoly je omezována použitou komunikační technologií.

Principy: STANDARDIZED CONTRACT.

Architektury: INVENTORY, SERVICE.

UML diagram:



Canonical Schema (Kanonické schéma)

Jak můžeme služby navrhnout, abychom se vyhnuli transformaci mezi datovými modely?

Problém: Služby s rozdílnými modely pro podobná data si vynucují dodatečné transformační požadavky, které zvyšují úsilí vynaložené při vývoji a návrhovou složitost a naopak snižují výkon aplikace za běhu kvůli dodatečné režii.

Řešení: Datové modely pro nejpoužívanější informační množiny jsou standardizovány ve všech kontraktech uvnitř inventáře.

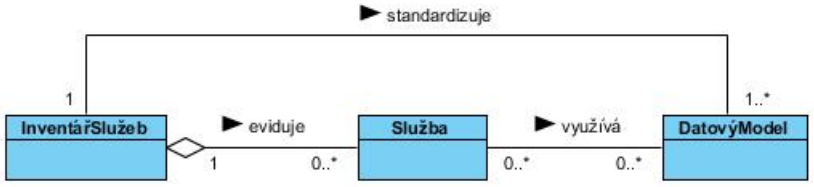
Aplikace: Návrhové standardy jsou aplikovány na schémata používaná servisními kontrakty jako část formálního návrhového procesu.

Dopady: Udržování standardizovaných schémat kontraktů může vyžadovat vysoké úsilí. Mohou nastat kulturní problémy.

Principy: STANDARDIZED CONTRACT.

Architektury: INVENTORY, SERVICE.

UML diagram:



3 Vzory pro tvorbu logických vrstev inventáře

Utility Abstraction (Abstrakce pomocné logiky)

Jak můžeme oddělit, opakovaně využívat a samostatně spravovat pomocnou logiku?

Problém: Pokud je veškerá logika vyžadovaná k automatizaci podnikového procesu obsažena v jediné službě, může dojít k redundantní implementaci běžných pomocných funkcí v mnoha službách.

Řešení: Je vytvořena servisní vrstva určená k provádění pomocných operací, která obsahuje znovupoužitelné pomocné služby využívané ostatními službami v inventáři.

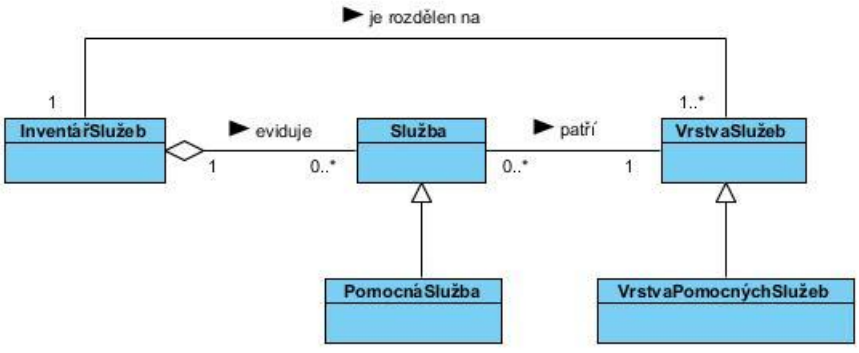
Aplikace: Model pomocných služeb je zahrnut do analytické a návrhové fáze procesů na podporu abstrakce pomocné logiky. Jsou podniknuty další kroky vedoucí k vybalancování kontextu služeb.

Dopady: Distribuování pomocné logiky mezi větší počet služeb může zvýšit požadavky na složitost a výkon.

Principy: LOOSE COUPLING, ABSTRACTION, REUSABILITY, COMPOSABILITY.

Architektury: INVENTORY, COMPOSITION, SERVICE.

UML diagram:



Entity Abstraction (Abstrakce procesně nezávislé logiky)

Jak můžeme oddělit, opakovaně využívat a samostatně spravovat procesně nezávislou logiku?

Problém: Spojováním procesně závislé a procesně nezávislé logiky do jediné služby může postupně docházet k vytváření redundantní agnostické logiky v mnoha službách.

Řešení: Vytvoříme procesně nezávislou vrstvu služeb věnovanou pouze službám, které mají funkční kontext založený na existujících podnikových entitách.

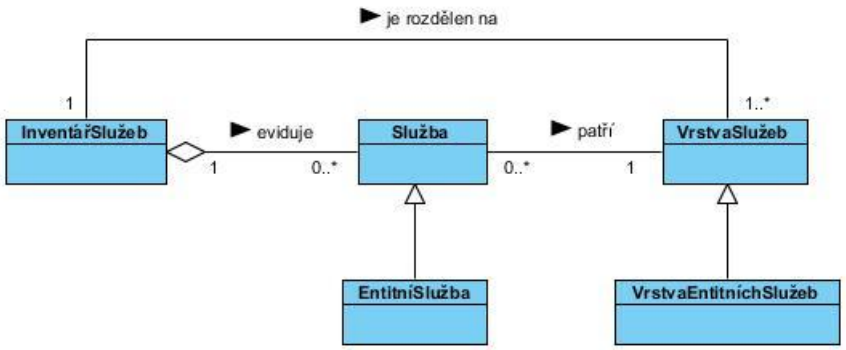
Aplikace: Kontext entitních služeb je založen na modelech obchodních entit a vytváří logickou vrstvu, která je modelována během analytické fáze.

Dopady: Podnikově zaměřená povaha těchto služeb vyžaduje zvýšenou pozornost při modelování a návrhu a požadavky na správu mohou přinést dramatické organizační změny.

Principy: LOOSE COUPLING, ABSTRACTION, REUSABILITY, COMPOSABILITY.

Architektury: INVENTORY, COMPOSITION, SERVICE.

UML diagram:



Process Abstraction (Abstrakce procesně závislé logiky)

Jak můžeme oddělit a samostatně spravovat procesně závislou logiku?

Problém: Spojováním procesně závislé a procesně nezávislé logiky dochází k problémům se správou procesně závislé logiky a znovupoužitím procesně nezávislé logiky.

Řešení: Vytvoříme vrstvu věnovanou procesně závislé logice, která podpoří nezávislou správu služeb jako potenciálních podnikových zdrojů.

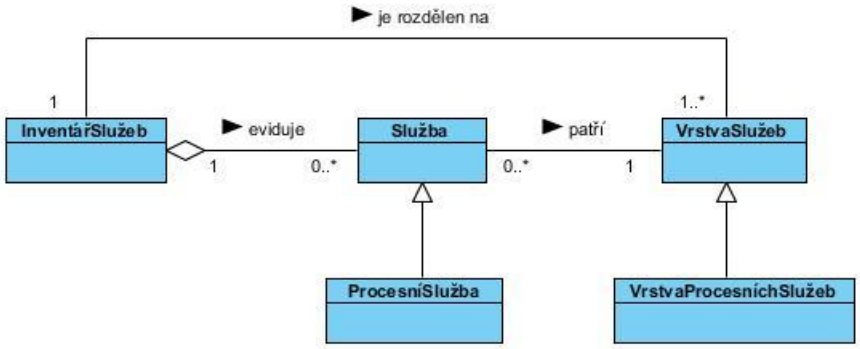
Aplikace: Jakmile jsou navrženy pomocné a entitní služby, vyfiltrujeme logiku obchodních procesů a definujeme procesně závislou servisní vrstvu.

Dopady: Kromě modelování a tvorby procesně závislých služeb, vytváří abstrakce rodičovské procesní logiky vrozenou závislost na provádění obchodní logiky pomocí kompozice dalších služeb.

Principy: LOOSE COUPLING, ABSTRACTION, COMPOSABILITY.

Architektury: INVENTORY, COMPOSITION, SERVICE.

UML diagram:



4 Vzory pro centralizaci inventáře

Process Centralization (Centralizace procesů)

Jak můžeme centrálně spravovat abstrahovanou logiku obchodních procesů?

Problém: Vyvíjení a rozšiřování systému může být problematické, pokud je logika obchodních procesů distribuována mezi více nezávislými implementacemi služeb.

Řešení: Logika reprezentující různé obchodní procesy může být spravována z jediného centrálního umístění.

Aplikace: Technologie potřebné k aplikování tohoto vzoru obvykle obsahuje dostupný middleware software.

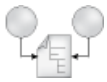
Dopady: Při zavádění těchto pomocných programů musíme počítat s podstatnými změnami infrastruktury a architektury.

Principy: AUTONOMY, STATELESSNESS, COMPOSABILITY.

Architektury: INVENTORY, COMPOSITION.

UML diagram:

Schema Centralization (Centralizace schématu)



Jak můžeme navrhnout kontrakty služeb, abychom se vyhnuli redundantní reprezentaci dat?

Problém: Různé kontrakty služeb často potřebují pracovat s podobnými obchodními dokumenty nebo datovými množinami, což vede k redundanci datových záznamů a obtížím s jejich správou.

Řešení: Vybrané datové modely existují jako fyzicky nezávislé části kontraktů služeb a jsou sdíleny vícero kontrakty.

Aplikace: Při předběžné analýze je třeba vytvořit nezávislou datovou vrstvu, která podporuje a je využívána vrstvou služeb.

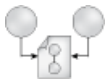
Dopady: Důležitost správy sdílených datových schémat roste s počtem služeb, které daná schémata využívají.

Principy: STANDARDIZED CONTRACT, LOOSE COUPLING.

Architektury: INVENTORY, SERVICE.

UML diagram:

Policy Centralization (Centralizace politik)



Jak můžeme předepsané politiky normalizovat a vynucovat jejich dodržování službami?

Problém: Politiky, které jsou aplikovány na vícero služeb, mohou způsobit redundanci a nekonzistenci uvnitř servisní logiky a kontraktů služeb.

Řešení: Globální nebo doménově specifické politiky můžeme izolovat a aplikovat na vícero služeb.

Aplikace: Opakovaně použitelné politiky by měly být definovány v analytické fázi a prosazovány pomocí rámce pro vykonávání politik.

Dopady: Rámce pro politiky snižují výkon a mohou záviset na soukromých technologiích. Je zde také riziko konfliktu mezi centralizovanými politikami a politikami specifickými pro danou službu.

Principy: STANDARDIZED CONTRACT, LOOSE COUPLING, ABSTRACTION.

Architektury: INVENTORY, SERVICE.

UML diagram:

Rules Centralization (Centralizace pravidel)

Jak můžeme abstrahovat a centrálně spravovat obchodní pravidla?

Problém: Stejná obchodní pravidla mohou být uplatňována v mnoha obchodních službách, což může vést k redundanci a problémům se správou.

Řešení: Centrum správy a úložiště obchodních pravidel je umístěno uvnitř dedikovaného architektonického rozšíření, ve kterém je možné k pravidlům centrálně přistupovat a spravovat je.

Aplikace: Systém správy obchodních pravidel je realizován pomocí systému agentů nebo dedikovaných služeb.

Dopady: Služby jsou vystaveny vyšším požadavkům na režii, větším rizikům a jsou více závislé na architektuře.

Principy: REUSABILITY.

Architektury: INVENTORY.

UML diagram:

5 Vzory pro implementaci inventáře

Dual Protocols
(Dvojí protokoly)

Jak může inventář služeb překonat omezení vyplývající ze vzoru Kanonický protokol, aniž by porušil pravidlo standardizace protokolu?

Problém: Vzor Kanonický protokol vyžaduje, aby všechny služby používaly jednotnou komunikační technologii. Avšak jediný komunikační protokol nemusí být schopen vyhovět požadavkům všech služeb.

Řešení: Architektura inventáře služeb je navržena tak, aby služby podporovaly primární a sekundární komunikační protokoly.

Aplikace: Jsou vytvořeny primární a sekundární úrovně služeb, které reprezentují koncovou vrstvu služeb. Všechny služby nadále podléhají doporučením vyplývajícím z návrhu služeb v servisně orientovaném paradigmatu a dopad porušení vzoru Kanonický protokol je dále snížen dodržováním specifických směrnic při návrhu služeb.

Dopady: Tento vzor může vést ke spletité architektuře inventáře služeb, zvýšeným nárokům na správu a (pokud je nesprávně aplikován) k nezdravé závislosti na vzoru Přemostění protokolů. Protože koncová vrstva není zcela jednotná, celkový počet potenciálních konzumentů a příležitostí ke znovupoužití služeb je snížen.

Principy: STANDARDIZED CONTRACT, LOOSE COUPLING, ABSTRACTION, AUTONOMY, COMPOSABILITY.

Architektury: INVENTORY, SERVICE.

UML diagram:

Canonical Resources (Kanonické zdroje)

Jak se můžeme vypořádat s nestejnorodostí externích zdrojů?

Problém: Implementace služeb mohou být závislé na různých externích zdrojích (databáze, adresářové služby apod.), čímž se zvyšují nároky na správu a údržbu služeb.

Řešení: Podpůrná infrastruktura a architektura může být vybavena společnými zdroji a rozšířeními, které mohou být opakovaně využívány různými službami.

Aplikace: Používání standardizovaných architektonických zdrojů je formalizováno definováním podnikových návrhových standardů.

Dopady: Pokud tento vzor vede k příliš vysoké závislosti na sdílených zdrojích, může dojít ke snížení autonomie a mobility služeb.

Principy: AUTONOMY.

Architektury: ENTERPRISE, INVENTORY.

UML diagram:

State Repository (Úložiště stavů)

Jak můžeme ukládat stavová data služeb, aniž bychom neúměrně zvyšovali nároky na systém za běhu?

Problém: Velké objemy stavových dat udržovaných za běhu programu pro potřeby běžících služeb mohou zabírat příliš mnoho paměti a snižovat tak škálovatelnost aplikace.

Řešení: Stavová data mohou být dočasně uložena do dedikovaného stavového repositáře a později z něj znovu načtena.

Aplikace: Sdílený nebo dedikovaný repositář dat je k dispozici jako součást inventáře nebo architektury služeb.

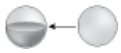
Dopady: Přidání vyžadované funkcionality pro čtení a zápis dat zvyšuje složitost návrhu služeb a může mít negativní vliv na výkon aplikace.

Principy: STATELESSNESS.

Architektury: INVENTORY, SERVICE.

UML diagram:

Stateful Services (Stavové služby)



Jak mohou být stavová data služeb trvale ukládána a spravována, aniž by docházelo k zatížení programu za běhu?

Problém: Stavová data náležející k určitým aktivitám služeb mohou na kompozice služeb vkládat příliš velké nároky ohledně správy stavů a snižovat tak škálovatelnost aplikace.

Řešení: Stavová data jsou ukládána a spravována v rámci k tomuto účelu určených pomocných stavových služeb.

Aplikace: Pomocné stavové služby udržují v paměti stavová data aktivních služeb.

Dopady: Pokud dojde k nesprávné implementaci, pomocné stavové služby mohou negativně ovlivnit výkon aplikace.

Principy: STALESSNESS.

Architektury: INVENTORY, SERVICE.

UML diagram:

Service Grid (Mřížka služeb)

Jak udržet stavová data služeb ve stavu co nejvyšší škálovatelnosti a odolnosti proti chybám?

Problém: Stavová data uložená ve stavovém repositáři nebo stavových službách mohou být zdrojem chyb nebo omezení výkonu aplikace, obvzláště jsou-li často používaná.

Řešení: Stavová data uložená v kolekci systémových stavových služeb zformují mřížku, která poskytuje vysokou škálovatelnost a odolnost vůči chybám díky replikaci paměti, redundanci a podpůrné infrastruktuře.

Aplikace: Do podnikové a inventářové architektury je zavedena gridová technologie.

Dopady: Aplikace tohoto vzoru může vyžadovat významné rozšíření infrastruktury a mít tak vyšší požadavky na správu.

Principy: STALESSNESS.

Architektury: ENTERPRISE, INVENTORY, SERVICE.

UML diagram:

Inventory Endpoint (Vstupní bod inventáře)

Jak může být inventář služeb ochráněn před nevhodnými vnějšími vstupy, aniž by byla ovlivněna jeho schopnost poskytovat služby vnějším konzumentům?

Problém: Skupina služeb v určitém inventáři může poskytovat funkcionalitu, která je užitečná službám mimo tento inventář. Z hlediska údržby a bezpečnosti však nemusí být vhodné vystavit všechny služby inventáře vnějším konzumentům.

Řešení: Požadovanou funkcionalitu abstrahujeme do koncové služby, která funguje jako oficiální vstupní bod dedikovaný určité skupině konzumentů.

Aplikace: Koncová služba může vystavovat servisní kontrakt s nabídkou funkcionalit, které poskytují služby v inventáři. Tyto funkcionality však mohou být dále pozměněny politikami a dalšími charakteristikami vhodnými pro komunikaci s konzumenty.

Dopady: Koncové služby mohou zvýšit svobodu při správě základních služeb inventáře, ovšem zároveň zvyšují požadavky na správu inventáře kvůli zavádění redundantní servisní logiky a kontraktů.

Principy: STANDARDIZED CONTRACT, LOOSE COUPLING, ABSTRACTION.

Architektury: INVENTORY.

UML diagram:

Cross-Domain Utility Layer (Mezidoménová pomocná vrstva)

Jak se můžeme vyhnout redundanci pomocné servisní logiky v několika doménových servisních inventářích?

Problém: V doménových inventářích služeb může docházet k redundanci ve vrstvě pomocných služeb.

Řešení: Založíme společnou vrstvu pomocných služeb pro několik doménových inventářů.

Aplikace: Množina společných pomocných služeb musí být definována a standardizována v koordinaci s vlastníky doménových inventářů služeb.

Dopady: Na koordinaci a správu mezidoménové vrstvy pomocných služeb je třeba vyvinout zvýšené úsilí.

Principy: REUSABILITY, COMPOSABILITY.

Architektury: ENTERPRISE, INVENTORY.

UML diagram:

6 Vzory pro správu inventáře

Canonical Expression (Kanonické vyjádření)

Jak zajistit, aby nedocházelo k nesprávnému porozumění a interpretaci servisních kontraktů?

Problém: Servisní kontrakty mohou vyjadřovat podobné schopnosti různými způsoby, což může vést k neporozumění a nesrovnalostem.

Řešení: Servisní kontrakty jsou standardizovány pomocí jmenných konvencí.

Aplikace: Jmenné konvence jsou na servisní kontrakty aplikovány jako část formálního procesu analýzy a návrhu.

Dopady: Zavedení globálních jmenných konvencí si vynucuje použití a neustálé dodržování podnikových standardů.

Principy: STANDARDIZED CONTRACT, DISCOVERABILITY.

Architektury: INVENTORY, SERVICE.

UML diagram:

Metadata Centralization (Centralizace metadat)

Jak mohou být metadata služeb centrálně publikována a spravována?

Problém: Projektové týmy, zejména ve velkých firmách, jsou neustále vystaveny riziku budování funkcionality, která již existuje nebo je ve vývoji. Toto zbytečně vynaložené úsilí vede k vytváření redundantní servisní logiky a denormalizaci inventáře služeb.

Řešení: Metadata služeb mohou být centrálně publikována v registru služeb a poskytovat tak formální prostředky k nalezení a registraci služeb.

Aplikace: Soukromý registr služeb musí být centrální částí inventářové architektury, aby mohl podporovat formální procesy nalezení a registrace služeb.

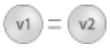
Dopady: Registr služeb musí být spolehlivý a kvalitní a jeho užívání a údržba musí být zahrnuta do všech procesů správy a dodávání služeb.

Principy: DISCOVERABILITY.

Architektury: ENTERPRISE, INVENTORY.

UML diagram:

Canonical Versioning (Kanonické číslování verzí)



Jak mohou být číslovány verze servisních kontraktů uvnitř inventáře služeb, aby nedocházelo k problémům s jejich správou?

Problém: Pokud jsou verze servisních kontraktů uvnitř inventáře služeb číslovány rozdílně, může docházet k problémům s jejich správou a interoperabilitou.

Řešení: Pravidla číslování verzí servisních kontraktů jsou standardizována pro celý inventář služeb.

Aplikace: K zajištění konzistence číslování verzí servisních kontraktů je třeba vyvinout a dodržovat návrhové standardy.

Dopady: Vytvoření a dodržování nutných návrhových standardů představuje zvýšené nároky na správu.

Principy: STANDARDIZED CONTRACT.

Architektury: SERVICE, INVENTORY.

UML diagram:

7 Závěr

Cílů práce nebylo prozatím dosaženo.

Literatura

[1] Erl, Thomas: SOA Design Patterns.
Upper Saddle River, Prentice Hall, 2008.