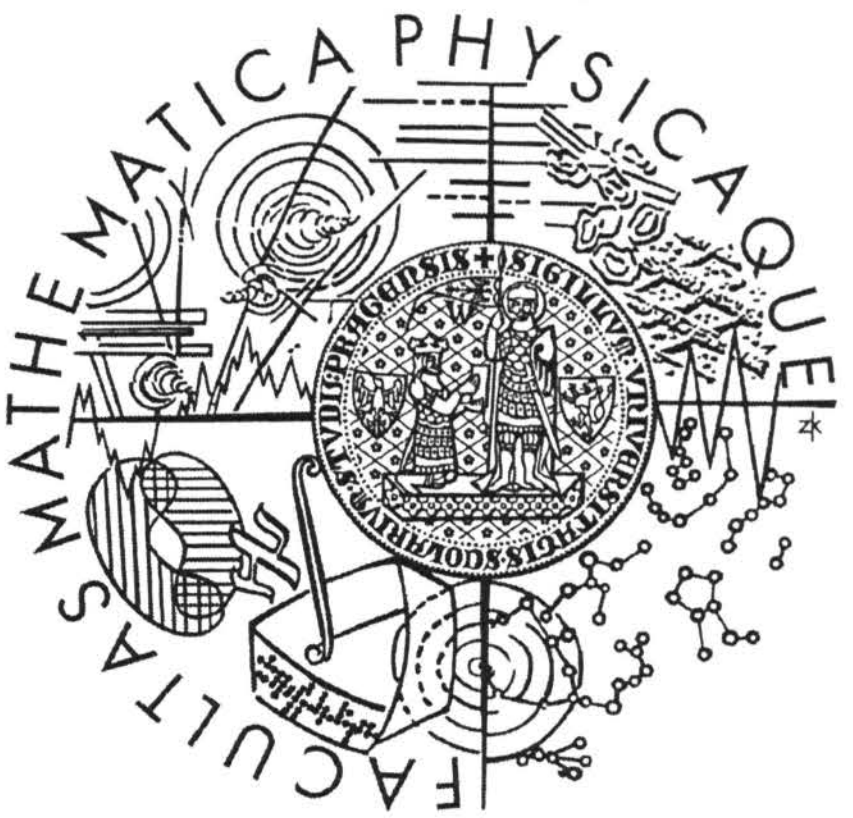
Univerzita Karlova v Prahe
Matematicko-fyzikálna fakulta

DIPLOMOVÁ PRÁCA



Ľudovít Dékány

Systémy založené na spolupráci služieb (SOA) používané mimo e-komerciu

Katedra softwarového inžinierstva
Vedúci diplomovej práce: Prof. RNDr. Jaroslav Král, DrSc.
Študijný program: Architektúra a princípy softwarových systémov

Poďakovanie

Chcel by som sa poďakovať Prof. RNDr. Jaroslavovi Královi, DrSc. za vedenie mojej diplomovej práce, za pomoc pri jej písaní a za poskytnutie hodnotných podkladov. Taktiež ďakujem spoločnosti Trask solutions s.r.o. za poskytnutie know-how ohľadom praktického využitia servisne orientovanej architektúry.

Prehlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičaním práce.

V Prahe dňa 7.8.2006

Ľudovít Dékány

Obsah

Zoznam obrázkov

Abstrakt

Názov práce: *Systémy založené na spolupráci služieb (SOA) používané mimo e-komerciu*
Autor: *Ľudovít Dékány*
Katedra (ústav): *Katedra softwarového inžinierstva*
Vedúci diplomovej práce: *Prof. RNDr. Jaroslav Král, DrSc.*
e-mail vedúceho: *Jaroslav.Kral@mff.cuni.cz*
Abstrakt: *Architektúra orientovaná na služby (SOA) je momentálne jedným z hlavných pojmov v oblasti informačných technológií. Princípy SOA sa postupne dostávajú do povedomia širokej IT verejnosti. Táto práca skúma fenomén SOA v prostredí mimo e-komerciu a skladá sa z dvoch komplementárnych častí. Prvá časť je orientovaná teoreticky a zameriava sa na objasnenie základných princípov SOA, ukazuje možné prínosy nasadenia a taktiež rizika nesprávneho prístupu. Detailne sú rozobrané i aspekty základného vzoru SOA, ktorým je Enterprise Service Bus. Druhá časť je zameraná skôr prakticky a skúma softwarový produkt Trask Integration Framework, ktorý vznikol v českých podmienkach, a aj napriek tomu je schopný konkurovať produktom kategórie Enterprise Service Bus nadnárodných spoločností. Práca objasňuje históriu a pozadie tohto produktu a detailne skúma produkt z pohľadu zvolených technológií, zvoleného prístupu a zvažuje tiež potenciálne prínosy a rizika pre zákazníka. Súčasťou práce je aj porovnanie produktu Trask Integration Framework s ďalšími produktmi dostupnými na trhu.*
Kľúčové slová: *SOA, ESB, TIF*

Title: *Service oriented architecture (SOA) used outside e-commerce*
Author: *Ľudovít Dékány*
Department: *Department of Software Engineering*
Supervisor: *Prof. RNDr. Jaroslav Král, DrSc.*
Supervisor's e-mail address: *Jaroslav.Kral@mff.cuni.cz*
Abstract: *Service oriented architecture (SOA) is one of the main buzzword used in information technology area these days. The principles of SOA are slowly getting well known to the wide IT community. This work explores the phenomenon called SOA outside e-commerce and is composed of two complementary parts. The first part is theoretically oriented and is focused on clarifying the basic principles of SOA; it shows the possible advantages of using SOA and also the risks when it is used incorrectly. The aspects of the basic SOA pattern, Enterprise Service Bus, are investigated in detail. The second part is oriented practically and explores the software product Trask Integration Framework (TIF). Although it was developed in Czech environment it is able to compete with other Enterprise Service Bus products from global companies. This work focuses on history and background of TIF; it also studies this product from the perspective of chosen technologies and approach and describes the potential assets and risks for customer. Part of the work is also a comparison of the Trask Integration Framework product with other products common on market.*
Keywords: *SOA, ESB, TIF*

1. Úvod

Service Oriented Architecture (SOA), architektúra orientovaná na služby, je podľa viacerých IT spoločností kľúčovým paradigma súčasnosti ale aj budúcnosti. Spoločnosti ako IBM, Microsoft, BEA, Sonic a mnoho ďalších sa preto predbiehajú v poskytnutí produktu, ktorý by umožnil efektívne, rýchle a jednoduché vytvorenie servisne orientovanej architektúry.

V čom je však čaro SOA, ktorému každý podlieha ?

Teoretická časť sa zameriava na popis servisne orientovanej architektúry použitej mimo e-komerciu a jej porovnanie i prepojenie s objektovou orientáciou. Zároveň sa detailne popisujú webové služby, ktoré sú v dnešnej dobe často mylne považované práve za servisne orientovanú architektúru. Práca popisuje taktiež modelové vzorové situácie, ktorých je treba sa pri používaní servisne orientovanej architektúry vyvarovať.

V závere sa popisuje architektonický štýl Enterprise Service Bus. Tento pojem rozširuje servisne orientovanú architektúru, nad ktorou definuje sadu Quality of Services. Tento architektonický štýl v dnešnej dobe implementujú vedúce spoločnosti IT charakteru.

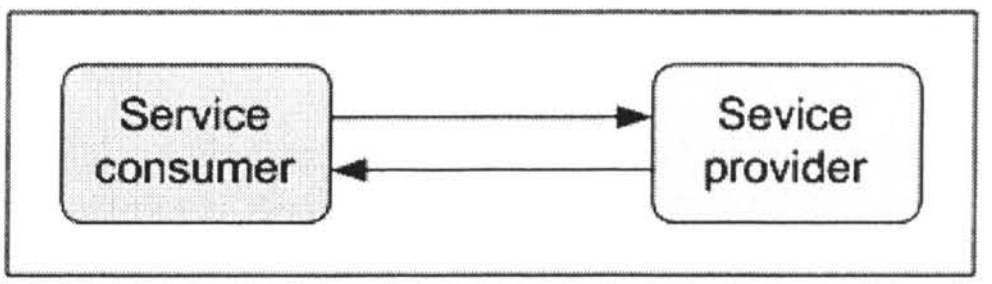
Praktická časť je zameraná na predstavenie produktu, ktorý dokáže konkurovať drahým produktom vedúcich IT spoločností dnešného sveta a zároveň poskytovať dostatočnú funkčnosť pre vytvorenie servisne orientovanej architektúry. Popisuje nielen samotnú architektúru produktu, ale aj príčiny vzniku a ideologické pochody pri návrhu a výbere konkrétnych technológií. Produkt je rozobraný nie len z perspektívy pozorovateľa a užívateľa, ale aj z perspektívy návrhára a vývojára, čím umožňuje čitateľovi nahliadnuť do výhod a nevýhod samotného produktu aj zo zákulisia.

2. Teória

2.1. Služba (Service)

Služba by sa dala definovať ako určitá časť autonómneho softwaru, implementujúca znovu použiteľnú funkčnosť. Služba má jasnú definíciu a je voľne dostupná určitej skupine spotrebiteľov. Služba i jej spotrebitelia tvoria koncové body virtuálnej P2P (Pier to Pier) siete. V rámci tejto P2P siete majú služba i jej spotrebitelia ekvivalentné postavenie. [1]

K pojmu služba sa viažu dva druhy rolí. Poskytovateľ služieb (service provider) a užívateľ (spotrebiteľ) alebo konzument služieb (service consumer). Poskytovateľ služby (softwarový komponent) je jedným koncovým bodom P2P siete. Služba je volaná za účelom uspokojiť potreby konzumenta služby. Konzument služby je druhým koncovým bodom P2P siete.

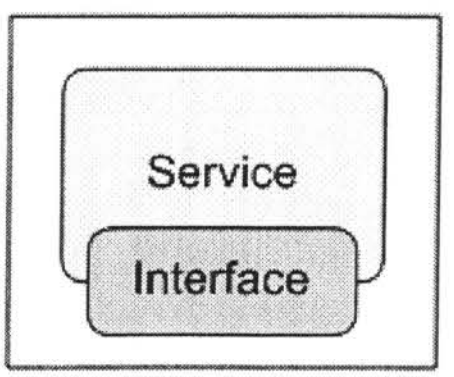


Obrázok 1 : Role viažuce sa k službe

Funkčnosť služby a tým aj službu samotnú definuje poskytovateľ služieb. Toto je dané častou anonymitou konzumentov a ďalej faktom, že služba je často verejná a jej konzumentom preto môže byť ktokoľvek.

Služba, ako jej názov už napovedá, je určitá netriviálna funkčnosť, ktorá je sprístupnená určitej skupine potenciálnych užívateľov. Konzumentov služby nie je možné konkrétne špecifikovať (kategorizovať). Konzumentom môže byť ktokoľvek, kto dokáže danú službu lokalizovať a využiť. Službu preto nemusí používať nutne len „obyčajná“ aplikácia. Konzumentom môžu byť napríklad aj iné služby alebo aj človek (napríklad prostredníctvom užívateľského rozhrania).

Služba ako časť autonómneho softwaru by sa dala rozčleniť na dve časti : rozhranie služby a implementácia služby.



Obrázok 2 : Služba a jej rozhranie

Dôležitejšou časťou služby z hľadiska celkového pohľadu na službu je jej rozhranie (interface). Rozhranie totiž definuje spôsob prístupu k službe a určitým spôsobom vymedzuje aj jej funkčnosť. Preto je v dobe návrhu služby veľmi dôležitá jasná predstava o budúcej funkčnosti vytváranej služby a formáte jej rozhrania. Vzhľadom na zdieľané používanie služby by bola neskoršia zmena rozhrania veľmi zložitá, a preto je nutnosť kvalitného návrhu rozhrania o to väčšia. Formát rozhrania služby je možno ešte dôležitejší ako samotný obsah rozhrania. Určitým spôsobom totiž vymedzuje cieľovú skupiny konzumentov služieb, ktorí sú schopný danému formátu rozhrania porozumieť.

Výber konkrétneho formátu rozhrania závisí od prostredia, v ktorom sa služby budú využívať.

Rozhrania v prostredí e-komercie (aliancie) sú väčšinou popísané a implementované pomocou štandardne definovaného jazyka. Výber formátu rozhrania môže zásadným spôsobom ovplyvniť použiteľnosť služby. Použitím neštandardného alebo propietárneho jazyka by sa okruh potenciálnych konzumentov služby radikálne znížil. Takúto službu by mohli využívať len konzumenti schopní popis služby pochopiť a porozumieť mu. V prostredí e-komercie ale nieje väčšinou dopredu známy okruh konzumentov služby. Tento okruh sa totiž počas životnosti služby môže ľubovoľne meniť.

Keďže rozhranie je jediná verejne prístupná časť služby, popisuje vlastne službu ako celok. Rozhranie musí preto obsahovať dostatočné množstvo informácií, ktoré službu identifikujú, aby ju prípadní konzumenti služby mohli nájsť, identifikovať, pochopiť a aj použiť. Tieto informácie sú o to dôležitejšie, že implementácia služby nie je verejne dostupná.

Rozhranie musí popísať všetky vstupné dáta a všetky výstupy služby, ako sú dátové toky alebo výnimky vyvolané počas vykonávania implementácie služby. Ďalej by malo dobré rozhranie obsahovať dostatok metadát pre popis služby, keďže názov služby nemusí dostatočne dobre popisovať službu samotnú, a taktiež môže existovať viac služieb rovnakého názvu. Názov služby síce o službe niečo vypovedá, ale v niektorých prípadoch môže byť názov len slabá identifikácia služby. U verejných služieb je kvalitný popis služby dôležitejší omnoho viac, než u služieb poskytovaných úzkemu okruhu konzumentov.

Formát rozhrania v prostredí mimo e-komerciu (konfederácia) však nie je nutne popísaný pomocou štandardne definovaného jazyka. V tomto prostredí existujú totiž väčšinou pevnejšie vzťahy medzi komunikujúcimi stranami. Preto sa v rámci konfederácií niekedy upúšťa od štandardne definovaného popisovacieho jazyka pre popis rozhraní a používajú sa proprietárne formáty správ.

Implementácia služby je zvyčajne považovaná za "čiernu skrinku", ktorá bližšie nešpecifikovaným spôsobom zaisťuje funkčnosť služby. Pre samotného konzumenta nie je dôležité ako je konkrétna služba naimplementovaná. Návrh implementácie služby je druhoradý, prvoradý je návrh rozhrania. Implementácia sa počas života služby môže podľa potrieb meniť, rozhranie by sa však meniť veľmi nemalo. Aj vďaka tomu, že implementácia služby je skrytá pred konzumentom, je prípadná zmena implementácie taktiež skrytá pred konzumentom a je čiastočne nezávislá na rozhraní služby. Implementácia služby môže byť jednoduchá v zmysle práce s jednoduchým objektom až po zložitú vnútornú logiku obsahujúcu volania iných služieb.

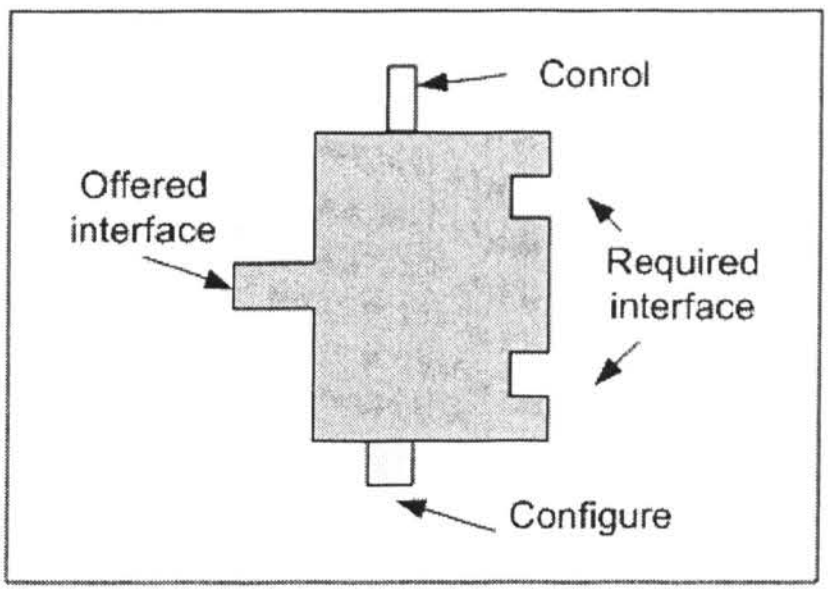
Ďalšia vlastnosť, ktorú však nie je možné nájsť v každej definícii služieb je bezstavovosť. Služby by nemali zdieľať stav medzi jednotlivými volaniami. To znamená, že implementácia

služby by nemala byť závislá na kontexte alebo stave predchádzajúceho volania služby. Táto vlastnosť nie je pre služby nutná, ale len doporučená.

2.2. Komponentová orientácia (Component orientation)

Komponentová orientácia je myšlienka vytvárania určitých funkčných jednotiek, znovu použiteľných stavebných blokov nazývaných komponenty. Neexistuje jednoznačná exaktná definícia komponentu.

Častá odkazovaná formulácia pojmu komponent od Clemensa Szyperski hovorí o softwarovom komponente ako o binárnej jednotke kompozície so zmluvne definovanými rozhraniami a explicitnými závislosťami. Komponent je nezávisle používaný tretími stranami.

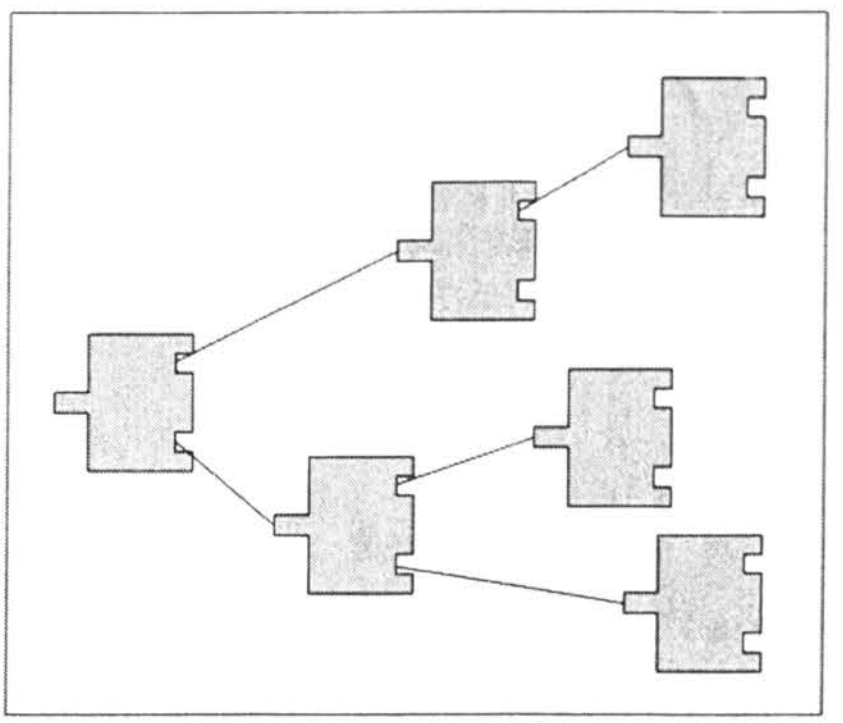


Obrázok 3 : Komponent

Táto formulácia hovorí o komponente ako o binárnej jednotke kompozície. Binárnej preto, lebo komponent je používaný tretími stranami a zdrojové kódy komponentov si ich tvorcovia chránia. Pre možnosť použitia však komponent musí poskytovať určitú sadu rozhraní, ktoré sprístupňujú funkčnosť tohto komponentu. Explicitnými závislosťami komponentu sa rozumejú externé konfigurácie vytvorené užívateľom komponentu za účelom určitého ovplyvňovania (dolaďovania) funkčnosti komponentu. Ďalšou možnou externou závislosťou môžu byť externé komponenty využívané daným komponentom pre svoju funkčnosť. Výhody komponentov zužitkováva kompozitná orientácia. [2][3]

2.3. Kompozitná aplikácia (Composite application)

Kompozitná aplikácia je aplikácia vytvorená za pomoci znovu použitia logiky z dvoch alebo viacerých už existujúcich komponentov bez nutnosti vytvorenia novej aplikácie od úplného začiatku. Kompozitná aplikácia využíva už implementovanú funkčnosť komponentov, ktoré je možné viacnásobne využívať. [4]



Obrázok 4 : Kompozitná aplikácia

Prečo sa ujala táto myšlienka využívania znovu použiteľných stavebných kameňov? Väčšina spoločností sa musí dynamicky prispôsobovať často sa meniacej situácii v oblasti ich pôsobenia. V e-kommercii to môže byť napríklad ponúkanie nových produktov, služieb, častá zmena obchodných parterov, tlaky konkurencie, v oblasti mimo e-kommercie to zase môže byť meniaca sa legislatíva alebo vývoj spoločnosti.

Na tieto, ale aj ďalšie rýchle zmeny musí reagovať takisto aj software, ktoré tieto spoločnosti využívajú k svojmu chodu. Vytvorenie takéhoto softwaru trvá určitý nezanedbateľný čas a prostriedky. Neustále vytváranie nového softwaru pri každej zmene by bolo prinajmenšom neefektívne. Preto je nanajvýš výhodné vytváranie softwaru z určitých ľahko znovu použiteľných stavebných kameňov.

Pre spoločnosti zaoberajúce sa výrobou softwaru je výhodné pri vytváraní tohto softwaru využiť už vytvorený kus kódu pre danú funkčnosť a nepísať implementáciu určitého problému opakovane. [5]

2.4. Objektová orientácia (Object orientation)

Objektová orientácia je myšlienka, pri ktorej sa objekty reálneho sveta zovšeobecňujú do objektov programovacích jazykov.

Všetky počítačové jazyky sú určitým spôsobom zovšeobecnením niečoho. Asembler zovšeobecňuje samotný počítač, na ktorom je daný kód asembleru spustený. Jazyky ako BASIC alebo FORTRAN zase zovšeobecňujú asembler.

Úlohou tohto zobecnenia je poskytnúť programátorovi určitý ‚svet', pomocou ktorého sa môže vyjadriť a pomocou ktorého nástrojov môže danú úlohu vyriešiť. Programátor vlastne vytvára určitú transformáciu sveta (priestoru) úlohy a sveta (priestoru) nástroja, v ktorom sa danú úlohu pokúšal riešiť. Vytváranie týchto transformácií vzhľadom na fakt, že odlišnosť dvoch koncových priestorov je relatívne veľká, bolo vo väčšine prípadov náročné. Vyžadovalo to určitú úroveň skúseností a výsledný kód bol často obsiahly a tým pádom aj ťažšie spravovateľný.

Objektová orientácia však priniesla priblíženie týchto dvoch koncových priestorov: priestoru úlohy a priestoru nástroja (jazyka).

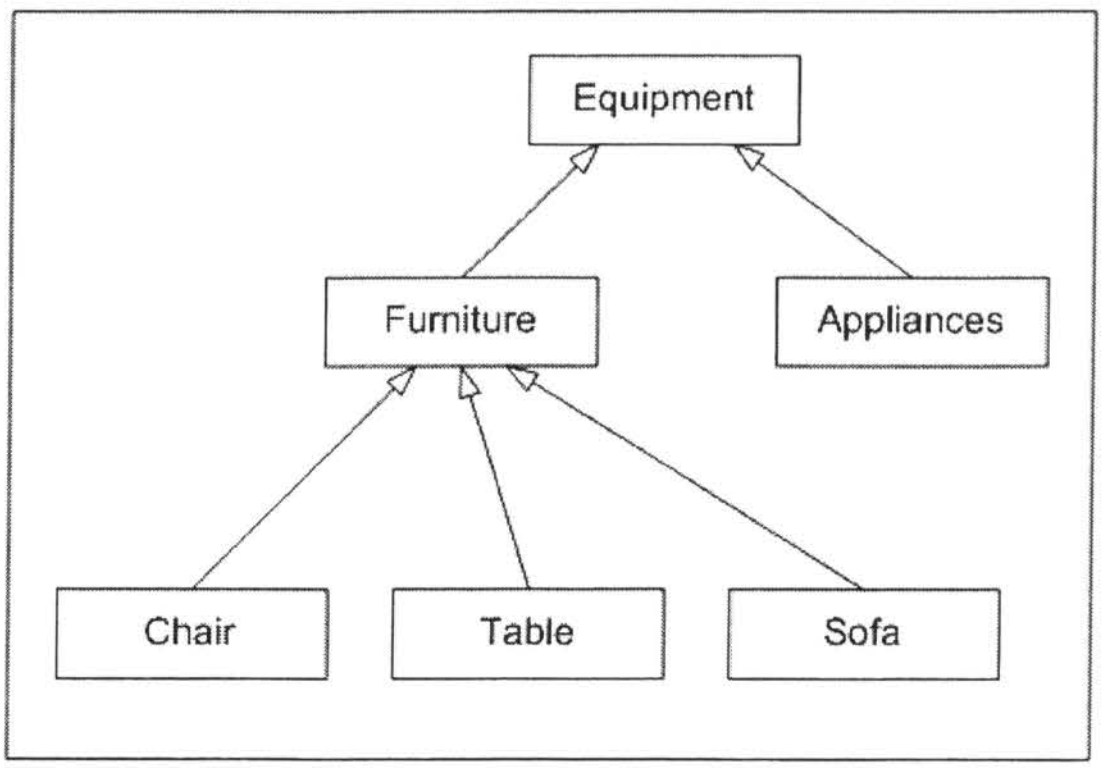
Objektová orientácia poskytuje nástroje na vyjadrenie ktoréhokoľvek prvku priestoru riešenia ako objektu priestoru nástroja bez nutnosti zložitej transformácie. Prvky úlohy sú abstrahované ako objekty, ktoré majú určité vlastnosti vyjadriteľné ďalšími objektmi a s týmito prvkami sa dajú prevádzať určité činnosti, ktoré sú v priestore riešenia vyjadrené metódami daných objektov.

Toto zobecnenie vďaka tomuto prístupu značne približuje oba koncové priestory, čo umožňuje nezanedbateľne urýchliť čas potrebný na riešenie väčšiny daných úloh. Výsledný kód je čitateľnejší a lepšie odráža riešený problém.

Aby sa s konkrétnymi objektmi pracovalo dobre, zavádza objektová orientácia určité stupne abstrakcie.
Konkrétnu stoličku (tú na ktorej práve sedím) je možné vyjadriť ako konkrétnu inštanciu objektu v priestore riešenia.

Všetky konkrétne objekty (inštancie) sa dajú zaradiť do určitých tried. Príkladom môže byť napríklad trieda stoličiek, ktorá pokrýva všetky stoličky rôznych vlastností (veľké stoličky, malé, modré alebo biele, dokonca aj tú moju, na ktorej práve sedím).

Tak ako aj v reálnom svete, kde sú všetky objekty (i triedy) reálneho sveta kategorizovateľné (stolička, polička i stolík sú všetky určitou podmnožinou skupiny objektov označovaných ako nábytok), bola zavedená určitá možnosť zoskupovania tried. Je teda možné triedu stolička vlastne považovať za konkrétnu podtriedu triedy nábytok, triedu nábytok ako podtriedu triedy zariadenie domácnosti atď. Vďaka tomuto prístupu je možné vytvoriť rôzne stromovité štruktúry pre popis reálneho sveta.



Obrázok 5 : Objektová orientácia

Ďalšou vlastnosťou objektov reálneho sveta je určité zdieľanie spoločných vlastností daných skupín tried, ktoré majú spoločnú nadtriedu. Táto vlastnosť je daná spôsobom akým sa jednotlivé nadtriedy v reálnom svete vytvárajú. Väčšinou sa totiž objekty (a tým aj triedy,

ktoré reprezentujú) dajú spojiť do jednej skupiny (nadtriedy) na základe spoločných vlastností.

Preto je pri definovaní tried možné triedam priraďovať vlastnosti (atribúty) a dokonca i metódy, ktoré odvodené triedy (podtriedy v strome tried) dedia. Príkladom môže byť vlastnosť farba u triedy zariadenie domácnosti, ktorú trieda nábytok dedí.

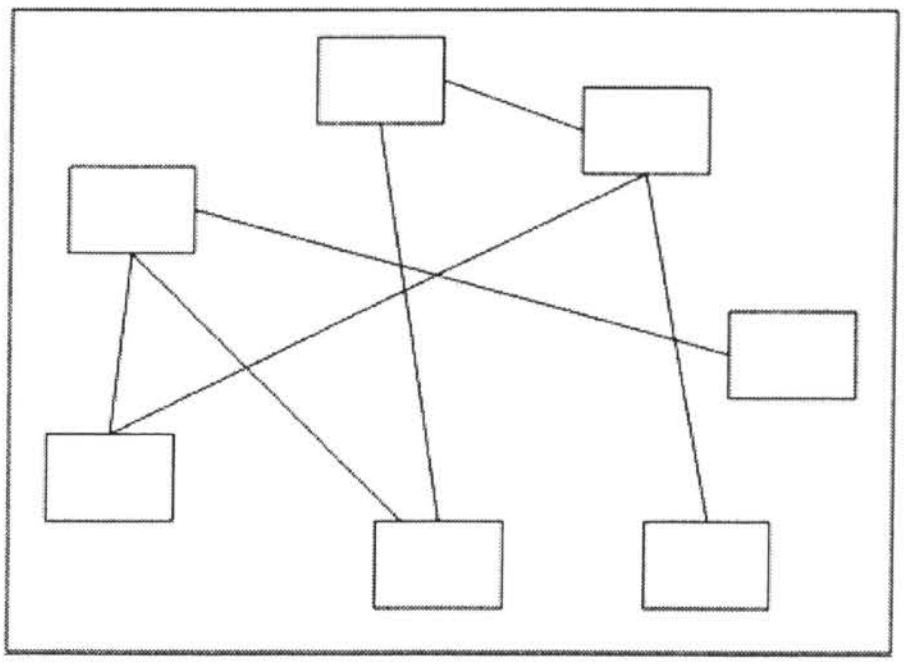
Ako bolo ukázané, objektová orientácia kopíruje obraz reálneho sveta a dokáže s kópiou tohto sveta pracovať. Komunikácia medzi dvoma objektmi prebieha pomocou volania metód, ktoré vracajú alebo menia dáta.

Hlavným rysom objektovej orientácie je relatívne vysoká jemnosť rozhrania. Objekty poskytujú veľké množstvo metód pre jemnú prácu s nimi. [6]

2.5. Servisná orientácia (Service orientation)

Servisná orientácia je myšlienka vytvárania softwaru za pomoci autonómnych komponentov tvoriacich virtuálnu P2P sieť. Tieto komponenty, ako naznačuje názov, sú tvorené službami (servismi). Verejne známou časťou služieb je ich rozhranie. Implementácia služieb je skrytá, tvorí akúsi ‚čiernu skrinku'.

Komunikácia v rámci tejto P2P siete prebieha pomocou asynchrónnej výmeny správ. Pri servisnej orientácii sa kladie vysoký dôraz na dynamickosť služby. Aplikácia pri vytváraní pozná len rozhranie volanej služby, ale implementácia služby sa môže počas životného cyklu aplikácie meniť. Ďalej je možná podpora dynamického vyhľadávania služieb, alebo výmeny služby za inú. [7]



Obrázok 6 : Servisná orientácia

2.6. Previazanosť (coupling)

Previazanosť (väzba medzi jednotlivými časťami, komponentmi softwaru) (coupling) je určité vyjadrenie závislosti medzi rôznymi komponentmi jedného systému. Inými slovami definovanie určitého stupňa závislosti jedného komponentu na druhom komponente. Existujú dva extrémy, čo sa previazanosti častí systému týka :

- Silná väzba (Tight coupling) – systémy s touto väzbou medzi jednotlivými časťami sú pevne zviazané a zmena jedného komponentu vyvoláva zmeny aj druhého komponentu.
- Slabá väzba (Loose coupling) – systémy s takouto väzbou majú nízky stupeň vzájomnej závislosti a je možné jeden komponent zmeniť bez následkov na druhý komponent.

U systémov so silnou väzbou sú komponenty previazané už v čase kompilovania, alebo v čase behu za pomoci synchrónneho volania. Zmeny prevedené do jedného komponentu sa prenášajú aj do ostatných komponentov práve kvôli silnej previazanosti. Táto silná previazanosť spôsobuje ťažšie testovanie i vývoj softwaru a prípadná zmena jedného komponentu je ťažko realizovateľná.

Výhody systémov so slabou väzbou:

- Možnosť dynamickej zmeny (alebo výmeny) komponentov počas životného cyklu systému. Táto zmena komponentu nemá vplyv na ostatné komponenty, a preto je táto zmena jednoduchá. Táto vlastnosť systémov so slabou väzbou je výhodná pre spoločnosti (organizácie), kde sa často menia určité časti systémov.
- Pri vytváraní komponentov systémov so slabými väzbami si vývojári môžu vybrať ľubovoľnú technológiu postavenú na ľubovoľnej platforme. Táto vlastnosť dovoľuje prepojenie rôznych subsystémov, ako sú proprietárne systémy napísané napríklad v jazyku Cobol slúžiace desiatky rokov s novšími jazykmi ako JAVA alebo C++.
- Takáto slabá previazanosť komponentov dáva určitú ochranu komponentom od seba navzájom. Z chyby jedného komponentu sa môže systém rýchlejšie zotaviť napríklad pomocou vymenenia inštancie jedného komponentu za inú inštanciu komponentu.

Na druhú stranu aj silná väzba má svoje využitie. Príkladom môžu byť komponenty, ktorých tesná previazanosť je kritická pre dizajn aplikácie.

Pri navrhovaní systémov preto nie je vhodné vyberať stupeň previazanosti jednotlivých komponentov na základe momentálnej popularity jedného či druhého prístupu. Malo by záležať skôr na požiadavkách vytváraného systému. Preto je dôležité, aby architekti (návrhári) správne pochopili výhody a dopady slabej i silnej väzby.

Dosiahnutie slabej väzby medzi jednotlivými komponentmi systému je v podstate vecou dodržania určitých zásad uvoľňujúcich väzbu medzi jednotlivými komponentmi. Medzi tieto zásady (patterns) patria :

- Nezávislosť umiestnenia komponentu – nie je podstatné kde sa jednotlivé komponenty systému nachádzajú. Možnosť jednoduchej zmeny komponentu dovoľuje využívať jeden deň napríklad jeden komponent umiestnený v rámci spoločnosti, ale druhý deň už využívať služby iného komponentu umiestneného na druhom konci sveta u partnerskej organizácie. Táto zásada je podporovaná vďaka možnosti dynamického vyhľadávania komponentov.
- Nezávislosť komunikácie – nie je podstatné aké komunikačné protokoly alebo formáty vytvárané/používané komponentom sa používajú k výmene správ. Pre prepojenie rozdielnych komunikačných protokolov však treba použiť štandardné protokoly pre prepojenie rôznych formátov.

- Nezávislosť bezpečnosti – nie je podstatné aké bezpečnostné modely jednotlivé komponenty používajú pokiaľ sa použijú štandardy, ktoré dokážu medzi jednotlivými bezpečnostnými modelmi takzvane prekladať.
- Nezávislosť inštancií – táto zásada hovorí o komponentoch, ktoré by mali podporovať synchrónnu i asynchrónnu komunikáciu a bezstavovosť komponentov. [8]

2.7. OO vs SO

Objektovo orientované technológie sú v dnešných časoch široko využívané a obľúbené. Toto paradigma je úž dávno overené a zažité. Existuje široká paleta nástrojov, ktoré túto technológiu podporujú.

Asi aj pre tieto skutočnosti objektovo orientované myslenie ovplyvňuje veľa ľudí používajúcich servisne orientované technológie. Pre presnejšie vyjadrenie ide o časté vytváranie služieb, ktorých rozhrania sú viac objektovo orientované než servisne orientované.

V podstate by to nebol žiaden veľký problém. Je možné vytvárať služby, ktorých rozhrania sú objektovo orientované. Túto orientáciu je možné spoznať podľa jemného rozhrania, ktoré poskytuje množstvo metód. Takto orientované rozhranie má však svoje obmedzenia.

Pre názorné porovnanie je vhodné načrtnúť jednotlivé možnosti vytvorenia kompozitnej aplikácie použitím rôzne orientovaných rozhraní medzi komponentmi. [9][10]

2.7.1. Objektovo orientované rozhranie

Pri objektovo orientovanom rozhraní je základným princípom využívanie zdieľaných objektov (zdieľané komponenty kompozitnej aplikácie). Pre úspešné použitie týchto zdieľaných objektov je ale nutné k týmto objektom získať odkaz (získať určitú referenciu, dokázať inštanciovať) v ostatných komponentoch, ktoré tento objekt využívajú. Preto je základným predpokladom v objektovo orientovanom systéme koncept spoločného zdieľania určitého typového systému. Tento systém je tvorený hierarchiou rozhraní, na ktorých sa komunikujúce strany musia dopredu dohodnúť. Až po dohodnutí sa na spoločných rozhraniach je možná integrácia objektovo orientovaného systému. Integrácia je následne dosiahnutá pomocou vzdialeného volania procedúr (Remote Procedure Call, RPC). Tieto objekty majú príliš veľkú granularitu a vzdialené volanie cez internet pridáva príliš veľké navŕšenie réžie. Navyše sú týmto riešením prepojené komponenty príliš zviazané. Organizácia spravujúca používaný komponent zmenením komponentu alebo jeho zmazaním naruší chod ostatných komponentov závislých na tomto komponente.

Určitým vylepšením systému založeného na objektovo orientovaných rozhraniach by bolo použitie technológií servisne orientovanej architektúry. Toto riešenie by eliminovalo nutnosť zdieľania určitého typového systému, keďže metadáta by boli uchovávané v určitom popisujúcom metajazyku (napr. WSDL). Toto vylepšenie umožňuje jednoduchšiu integráciu rôznych platforiem či rôznych programovacích jazykov.

Takýmto vylepšením objektovo orientovaného rozhrania však ostávajú niektoré deklarované nevýhody. Granularita rozhrania sa nezmenila a réžia transformovania volaní do XML správ a následné spätné transformovanie správy do volania funkcie cieľového komponentu ostáva. [9][10]

2.7.2. Servisne orientované rozhranie

U servisne orientovaného rozhrania je základnou myšlienkou vytvorenie služieb, ktoré poskytujú určité business funkčnosti (ucelená funkčnosť pokrývajúca určitú business operáciu). Tieto operácie majú charakter jednej komplexnej operácie využívajúcej k svojej činnosti viacej elementárnych operácií, alebo aj volanie iných služieb. Takto definované rozhrania majú väčšiu granularitu. Počet vystavených operácií i počet volaní zdieľaného komponentu sa zníži, čo má za dôsledok zníženie réžie pri volaní služieb. Závislosť komponentov používajúcich servisne orientované rozhrania sa zníži a zmena komponentu nemá taký dopad na ostatné komponenty ako je to pri objektovo orientovaných rozhraniach. Vnútorná implementácia komponentu sa môže ľubovoľne meniť, pričom však rozhranie zostáva rovnaké. Platformová alebo jazyková nezávislosť je tiež zaručená. [9] [10]

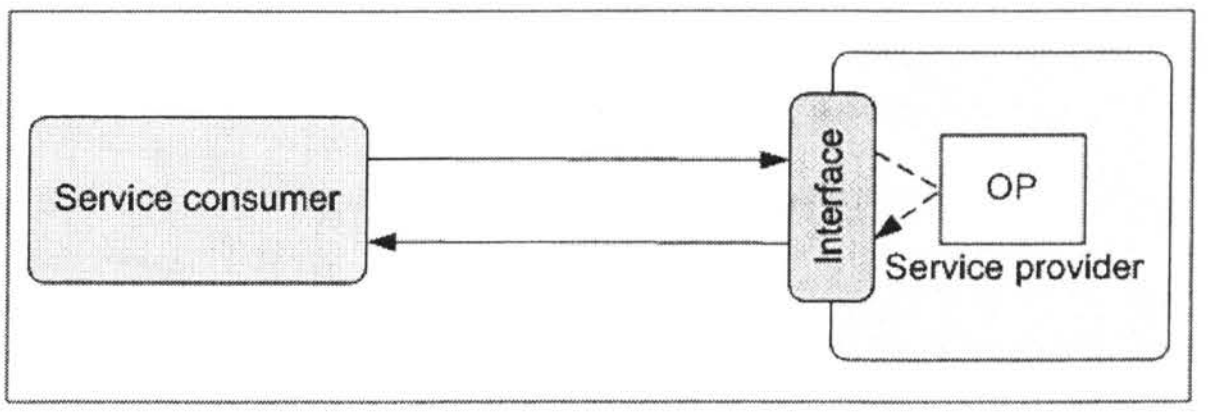
2.8. Charakter vzájomného posielania správ

Charakter vzájomného posielania správ medzi dvoma komunikujúcimi komponentmi sa dá obecne rozdeliť do dvoch skupín.

- Procedurálne orientované
- Orientované na dokumenty

Procedurálne orientované správy sa používajú u služby, ktorá má charakter funkcie s definovanými vstupnými a výstupnými parametrami. Dôležitým predpokladom pre tieto funkcie je ich odozva v reálnom čase. V implementácii funkcie sa na požiadavku v reálnom čase vygeneruje odpoveď a tá sa posiela späť konzumentovi služby. Toto posielanie správ má charakter synchrónnych správ. Typickým príkladom môže byť prístup do vzdialenej databáze alebo "real-time" služba pre objednávku určitého tovaru.

Objektovo orientované rozhrania sú väčšinou procedurálne orientované.



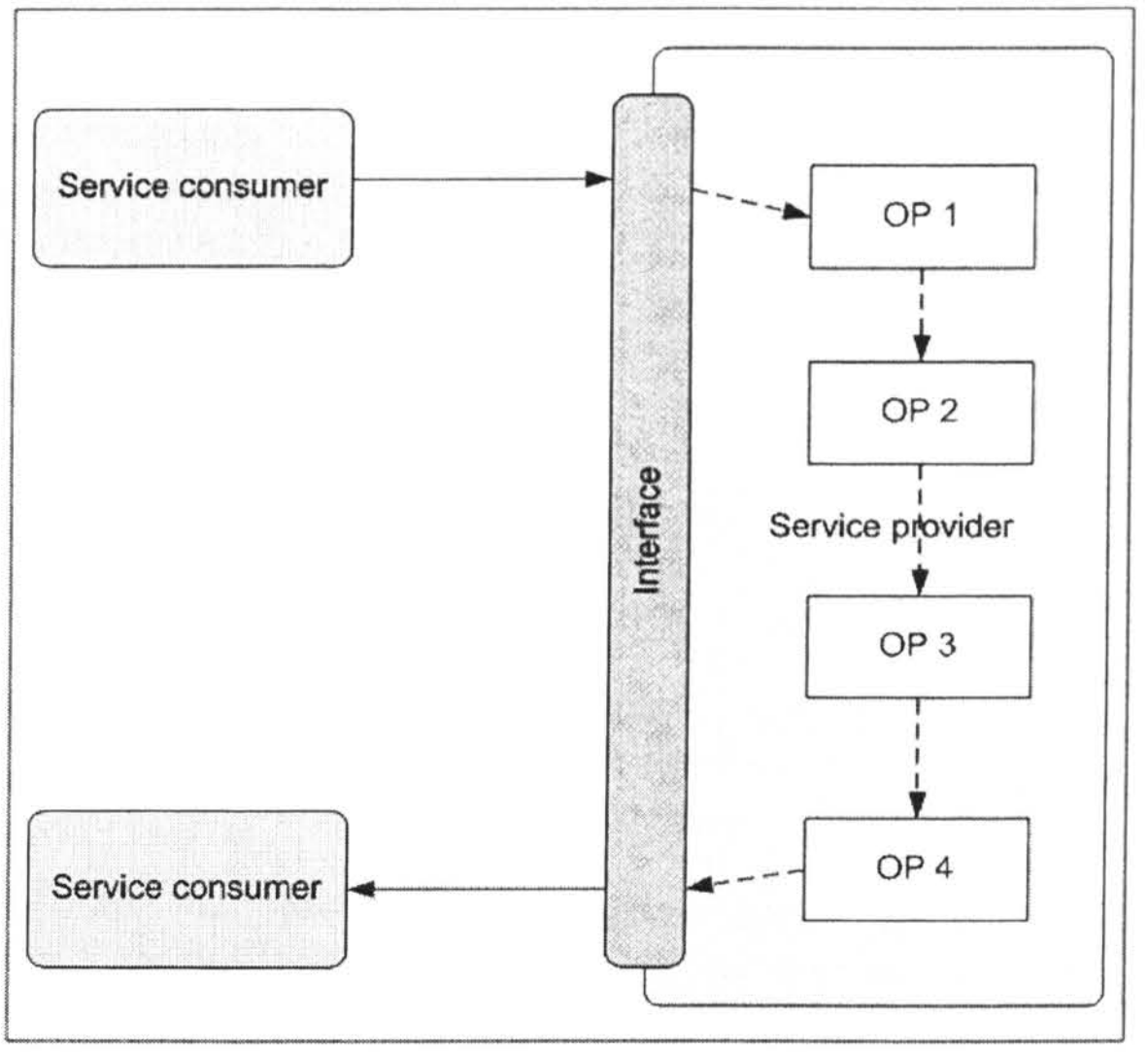
Obrázok 7 : Procedurálne orientované posielanie správ

U dokumentovo orientovaných správ má požiadavka na službu formu komplexnej správy obsahujúcej dostatočné množstvo informácií potrebných na vykonanie zložitej operácie, ktorá sa bude spracovávať ako celok. Príkladom by mohla byť zložitá objednávka daného tovaru, ktorá vyžaduje viac netriviálnych krokov. Medzi tieto kroky môže patriť zistenie určitých údajov z databáze, poprípade spracovávanie zložitých výpočtov, alebo aj služieb iných dodávateľov (partnerov). Keďže spracovanie takejto správy nemusí skončiť v reálnom čase, zvykne poskytovateľ služby poslať nejakou formou konzumentovi služby oznámenie o prijatí správy, poprípade sa posielajú aj pravidelné informácie o stave spracovávania správy.

Samozrejme pri používaní takýchto služieb musí mať konzument služby predstavu o funkčnosti služby. Zvyčajne sa takéto správy vyskytujú medzi obchodnými partnermi, ktorí sú dohodnutí na konkrétnejšej interakcii medzi sebou počas spracovávania správy, počínajúc oznamovacími emailami o stave spracovania správy, cez rôzne súhlasné správy o vývoji objednávky/príkazu až po posielanie účtov alebo peňazí medzi partnermi.

Posielanie dokumentovo orientovaných správ má charakter asynchrónneho posielania správ do fronty požiadaviek, často bez požadovania odpovede.

Dokumentovo orientované správy sú väčšinou doménou servisne orientovaných rozhraní.



Obrázok 8 : Dokumentovo orientované posielanie správ

2.9. Servisne orientovaná architektúra (Service oriented architecture) SOA

Servisne orientovaná architektúra je architektonický štýl, ktorého cieľom je vytvorenie slabej väzby medzi službami poskytujúcimi servisne orientované rozhrania. SOA definuje len architektúru, nedefinuje konkrétne technológie pomocou ktorých je možné servisnú orientáciu dosiahnuť. [11]

Definícia servisne orientovanej architektúry od IBM sa zakladá na webových službách ale nevylučuje i použitie iných technológií. [12]

Vytvorenie servisne orientovanej architektúry je možné dodržaním niekoľkých zásad [13]:

- Autonómnosť – služby sú autonómne

- Zdieľanie formálneho popisu – pre vzájomnú komunikáciu je potreba zdieľať formálny popis služby umožňujúci volanie služby
- Slabá väzba – väzba medzi poskytovateľom a konzumentom služby je slabá
- Skladateľnosť – služby môžu volať ďalšie služby čím vznikajú rôzne úrovne abstrakcie služieb
- Znovupoužiteľnosť – služby sú vytvárané za účelom vytvorenia znovupoužiteľných funkčností
- Bezstavovosť – služby by mali byť maximálne bezstavové
- Zistiteľnosť – služby by mali umožňovať vyhľadávanie svojich popiskov pre umožnenie používania služby

Medzi výhody SOA patria [14]:

- Flexibilná architektúra
- Nižšie náklady na integráciu
- Zvýšená znovupoužiteľnosť zdrojov
- Vysoká návratnosť investícií

2.10. SOA antipatterns

Servisne orientovaná architektúra je síce vynikajúca myšlienka vytvárania systémov orientovaných na služby, ktoré sú ľahko rozšíriteľné, lacné a vynikajú ešte mnohými výhodami, ale pre vytvorenie SOA je potrebné vyvarovať sa určitých situácií.

Vzor (pattern) je zachytenie problému a jeho úspešného riešenia. Slúži na predanie znalostí a skúseností nadobudnutých riešením daného problému. Tento príklad by mal poučiť ľudí znalých vzoru o možnosti identifikácie a riešenia problému.

Anti vzor (antipattern) je vzor poukazujúci na nesprávne riešenie, ktorého sa treba vyvarovať. Poukazuje na situáciu, vlastnosti riešenia ktoré sa uberá nesprávnym smerom. Podáva spôsob ako dané riešenie dostať znova späť na správnu cestu.

Nasledujúce SOA antipatterns popisujú modelové situácie, ktorých je treba sa vyvarovať pri implementovaní servisne orientovanej architektúry.

SOA antipatterns :

- Technology bandwidth
- So, What's New?
- The Big Bang
- Web service = SOA
- The Silo Approach
- Misbehaving Registries
- Chatty Services
- Point-to-point Services
- Component-less Services

Technology bandwidth – adaptovanie SOA len na technologickej úrovni ale nie na business úrovni. Následkom sú zvýšené náklady na IT bez výraznej návratnosti investícií. Ako riešenie tohto problému je vytvorenie benefitov SOA získateľných riešením.

So, What's New? – situácia kedy skeptici v spoločnosti argumentujú proti SOA tvrdením, že SOA je len nový názov pre niečo, čo je už v spoločnosti implementované. Následkom je neimplementovanie SOA a strata možnosti čerpať jeho výhod. Riešením je porovnanie existujúceho riešenia so SOA a zvýraznenie prínosov.

The Big Bang – situácia opačná problému So, What's New?. Prevláda presvedčenie presadiť SOA okamžite a čo najrýchlejšie. Neuváženým aplikovaním SOA sa žiadaný efekt nedostaví ale naopak, môže to viesť k návratu k starému riešeniu a zanevreniu na SOA. Riešením je vytvorenie určitého postupu postupného nasadzovania SOA s prihliadnutím na business výhody.

Web service = SOA – prevláda mylná predstava, že webové služby sú synonymom pre SOA. Nahradenie existujúcich API volaním webových služieb bez odpovedajúcej architektúry môže viesť k zahlteniu webovými službami bez výrazného efektu. Riešením je vytvorenie plánu prechodu na SOA a vzdelávanie v oblasti SOA a webových služieb.

The Silo Approach – problém identifikovania jednotlivých služieb založený na identifikovaní služieb podľa izolovaných aplikácií a nie na základe business funkčnosti. Následne to vedie k duplikovaniu tých istých služieb pod iným názvom a tento fakt navyšuje investície a znižuje ziskovosť. Riešením je vytvorenie spoločného frameworku pre identifikovanie služieb.

Misbehaving Registries – problém vznikajúci duplikovaným nesprávnym používaním centrálnych registrov služieb. Následkom môžu byť problémy identifikovania služieb napríklad pri vyhľadávaní chýb. Ďalším dôsledkom môže byť znížená bezpečnosť a výkonnosť. Riešením je adaptácia „best practices" používania centrálnych registrov.

Chatty Services – problém vznikajúci implementovaním služieb s rozhraním vysokej granularity. Výsledkom je veľká výmena dát a príliš časté volanie služieb, čo má za následok stratu výkonnosti a zvýšené náklady na vývoj. Riešením je vytváranie služieb s business orientovaným rozhraním.

Point-to-point Services – problém nahradenia integračného riešenia P2P volaním webových služieb bez ohľadu na skutočné použitie riešenia. Následkom je strata benefitov SOA riešenia. Riešením tohto problému je použitie podnikovej zbernice (Service Bus) pre vytvorenie SOA implementácie.

Component-less Services – problém vytvárania služieb bez jasného priradenia služby k danému komponentu. Vytvárajú sa služby bez vrstvenia. Následkom je porušenie princípov modulárneho designu a princípov logickej štruktúry, čo má za následok stratu flexibility. Riešením je uvedomenie si princípov SOA. [15]

2.11. Webové služby (Web services)

V podstate neexistuje jedna exaktná definícia webových služieb. Na druhú stranu väčšina definícií sa dá zaradiť do jednej z dvoch skupín.

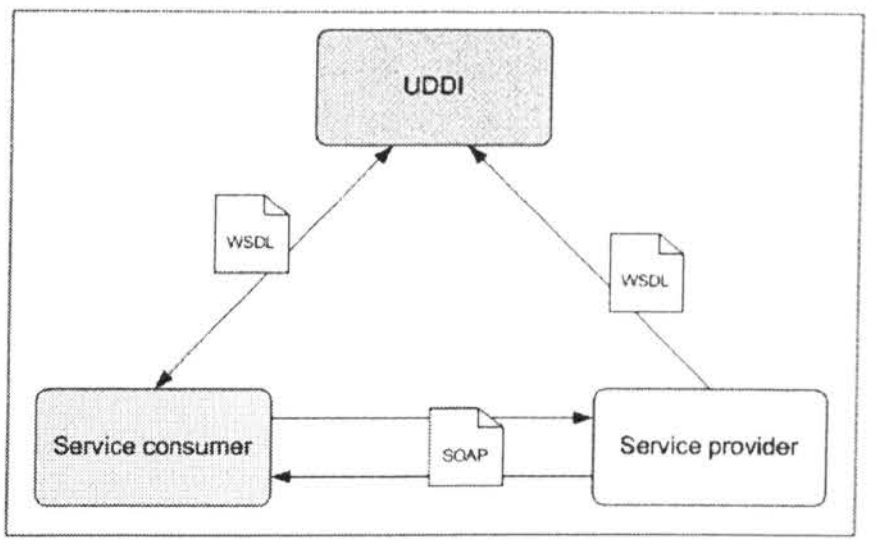
Jedna skupina definícií hovorí o webových službách ako o sade doporučených technológií pomocou ktorých je možné vybudovať servisne orientovanú architektúru. Základnými stavebnými kameňmi (komponentmi) tejto architektúry sú služby. Tieto služby sú prepojené navzájom práve pomocou webových služieb. Sadou technológií sa myslia SOAP, WSDL a UDDI. [16]

Druhá skupina definícií by sa dala zhrnúť ako definícia hovoriaca o webových službách ako o aplikáciách založených na technológiách SOAP, WSDL a UDDI. Tieto definície teda Webovými službami nemyslia sadu technológií, ale už konkrétne implementované služby volané pomocou protokolu SOAP. [17]

Definícia Webových služieb od IBM je obdobná ako prvá definícia popisujúca Webové služby ako sadu technológií. IBM sa však od tejto definície odchýlila tým, že vyslovene nehovorí o UDDI, ale len o SOAP a WSDL. Nejde o to, že by IBM UDDI nepovažovalo za jednu z technológií Webových služieb, len prevláda názor že UDDI nie je tak dôležité ako ostatné dve normy. [18]

Trojica technológií webových služieb slúži na :

- SOAP – definícia formátu správy pomocou ktorých služby navzájom komunikujú
- WSDL – definícia formátu popisujúceho rozhranie služby
- UDDI – popisuje centrálny register definícií služieb.



Obrázok 9 : Technológie webových služieb

Pri použití webových služieb dokážu objekty, programy, databáze od jednoduchých až po rozsiahle pomocou vytvoreného SOAP dokumentu poslať správu (požiadavka) nejakej službe a poprípade takisto vo forme SOAP dokumentu dostať požadovanú odpoveď. Webové služby práve definujú formát týchto správ, definujú popis rozhrania služby, ktorá bola volaná, definujú mapovanie obsahu správ z a do programovacieho jazyka, v ktorom je program napísaný a takisto definujú spôsob zverejňovania popisku služby a jej následné vyhľadávanie.

Možné použitie webových služieb je rozsiahle. Môžu byť použité v aplikačných klientoch na prístup do rezervačných alebo informačných služieb. Webové služby môžu byť napríklad

použité aj v systémoch B2B (business to business) pre systém integrujúci obchodný proces rôznych spoločností v jednom systéme, alebo napríklad aj v systémoch EAI (enterprise application integration) pre systém spojujúci rôzne podnikové systémy viacerých podnikov za účelom vzájomnej spolupráce.

2.12. WSDL – Web Services Description Language

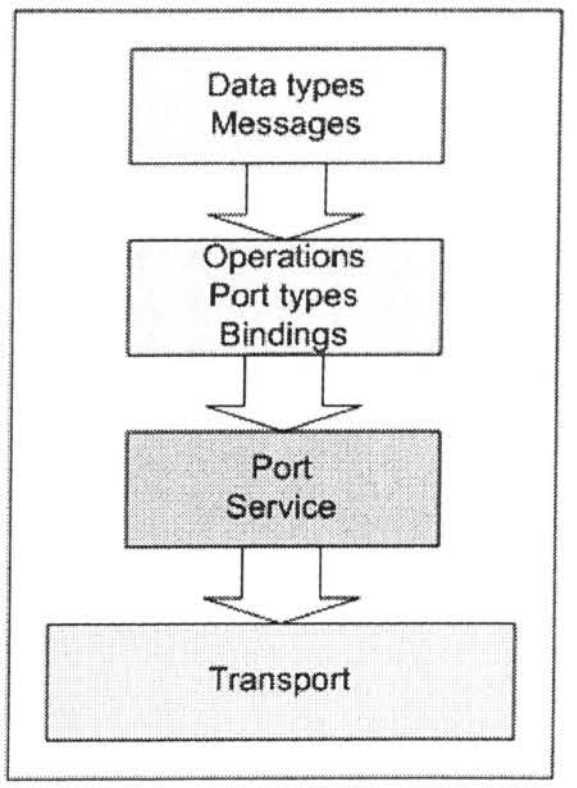
Názov WSDL by sa dal preložiť ako jazyk pre popis webových služieb. Tento názov dobre popisuje, na čo by WSDL vlastne malo slúžiť. WSDL je formát pre popísanie rozhrania webových služieb. WSDL poskytuje spôsob ako popísať rozhranie webovej služby i jej viazanosť na určitú sieťovú adresu. Vzniknutý WSDL dokument má formát XML dokumentu.

WSDL vzniklo za spolupráce spoločnosti Microsoft a IBM. K predloženiu na W3C sa pripojilo ďalších 23 spoločností.

WSDL spolu s ďalšími formátmi tvorí jadro Webových služieb. Je učené pre oba typy orientácie správ, tak ako procedurálne, tak i dokumentovo orientovaných.

WSDL sa skladá z troch základných častí :

- Definície typov
- Definície operácií
- Definície viazanosti služieb



Obrázok 10 : Časti WSDL

Jednotlivé časti môžu byť popísané v samostatných súboroch a neskôr rôzne zoskupované do výsledných WSDL súborov. Táto možnosť zaručuje možnosť opakovaného používania napríklad definícií alebo používanie globálnych definícií v rámci organizácie. Pre jednoduchosť WSDL súboru ale môžu byť všetky časti aj v jednom súhrnnom súbore.

Štruktúra WSDL súboru je rozdelená na tri úrovne v tejto podobe zámerne. Každá z častí je vlastne iná úroveň abstrakcie. Tieto úrovne abstrakcie sú definované nezávisle na transportnej vrstve tak, aby bolo možné definovať zároveň i niekoľko transportných vrstiev pre jednu službu. Napríklad je možné definovať službu prístupnú pomocou SOAP cez HTTP alebo ako SOAP cez JMS.

Časť definície typov sa skladá z dvoch častí : definície dátových typov (types) a definície správ (messages). V časti types sa definujú dátové typy potrebné pre posielanie správ. Pre maximálnu prenosnosť sa u WSDL používa XSD kanonický systém typov a je považovaný za východzí systém dátových typov. Pri definícii je možné definovať nový pomenovaný dátový typ použiteľný v celom vytváranom WSDL dokumente, poprípade vo všetkých dokumentoch, do ktorých sa táto definícia zahrnie.

Definícia správ definuje správy a ich jednotlivé typy pomocou dátových typov. Každá správa má jednoznačné meno, môže sa skladať z niekoľkých pomenovaných častí, ktoré môžu mať svoje dátové typy. Názov správy je unikátny medzi správami dokumentu, podobne je názov časti správy unikátny v rámci jednej správy. Definícia správy je zámerne takto obecná, keďže správa môže byť neskôr naviazaná viac druhov konkrétnych formátov.

Definície operácií je možné rozložiť na tri menšie časti : definícia operácií, definícia typov portov a definícia väzieb.

```
<?xml version "1.0" encoding "UTF-8"?>
<definitions>
    <types>
      .
      .
    </types>

    <message>
        <part/>
    </message>
    <message>
        <part/>
    </message>

    <portType>
        <operation>
            <input/>
            <output/>
            <fault/>
            <fault/>
        </operation>
    </portType>

    <binding>
        .
        .
        <operation>
            <input>
                <body/>
            </input>
            <output>
                <body/>
            </output>
        </operation>
    </binding>

    <service>
        <port>
            <address/>
        </port>
    </service>

</definitions>
```

Obrázok 11 : WSDL

WSDL podporuje možnosť odpovede na správu a preto sú v ňom definované 4 typy operácií :

- One way – jednosmerná správa od konzumenta k poskytovateľovi, obsahuje len input.
- Request-response – poskytovateľ služby dostáva správu, na ktorú následne odpovedá správou, obsahuje input, následne output a poprípade ľubovoľný počet fault.
- Solicit-response – poskytovateľ odosiela správu a následne dostáva správu od konzumenta, obsahuje output, následne input a poprípade ľubovoľný počet fault.
- Notification – poskytovateľ pošle správu konzumentovi, obsahuje len output

Input, output i fault sa odkazujú na už definované správy. Názov inputu a outputu nemusí byť definovaný, naopak fault musí mať meno zadané. Zadané mená musia byť jednoznačné v rámci jedného portType.

Operácie sú uzavreté do tagu portType, ktorý definuje logicky späté operácie, operácií môže byť viac, musia ale byť jedinečného mena.

Definícia väzieb (binding) definuje pre portType formát správy a detaily protokolu. Pre jeden portType môže byť ľubovoľné množstvo väzieb. Každá väzba má jednoznačne definované meno v rámci WSDL dokumentu. Element binding umožňuje pre každý portType, pre každú operáciu portTypu, a pre každý input, output a fault správu definovať rozširujúce informácie obsiahnuté ako elementy v týchto tagoch.

Posledná časť WSDL, definícia viazaností služieb, sa skladá z dvoch častí : definície portov a definície servisov.

Definície portov (ports) priraďujú jednotlivým väzbám (binding) koncové body priradením adresy.

Definícia služieb (service) zoskupuje logicky spojené porty do skupín. Samozrejme názov servisu i portu sú unikátne v rámci celého WSDL dokumentu. [19]

2.13. SOAP – Simple Object Access Protocol

Technológia SOAP je definovaná ako protokol pre vytvorenie „Messaging frameworku", tzn. frameworku založenom na správach.

SOAP verzia 1.2 je protokol určený pre výmenu štruktúrovaných dát v decentralizovanom, distribuovanom prostredí. Vďaka tomu, že je založený na technológii XML je možné pomocou neho vytvoriť jednoducho rozšíriteľný framework pre výmenu správ nad širokým spektrom transportných vrstiev (protokolov). Tento framework je následne nezávislý na programovacích modeloch, programovacích jazykoch alebo platformách.

Hlavnými cieľmi SOAP sú jednoduchosť a rozšíriteľnosť. Tieto ciele sú dosiahnuté „vynechaním" určitých vlastností, ktoré poskytujú distribuované systémy. Medzi tieto vlastnosti patria napríklad :

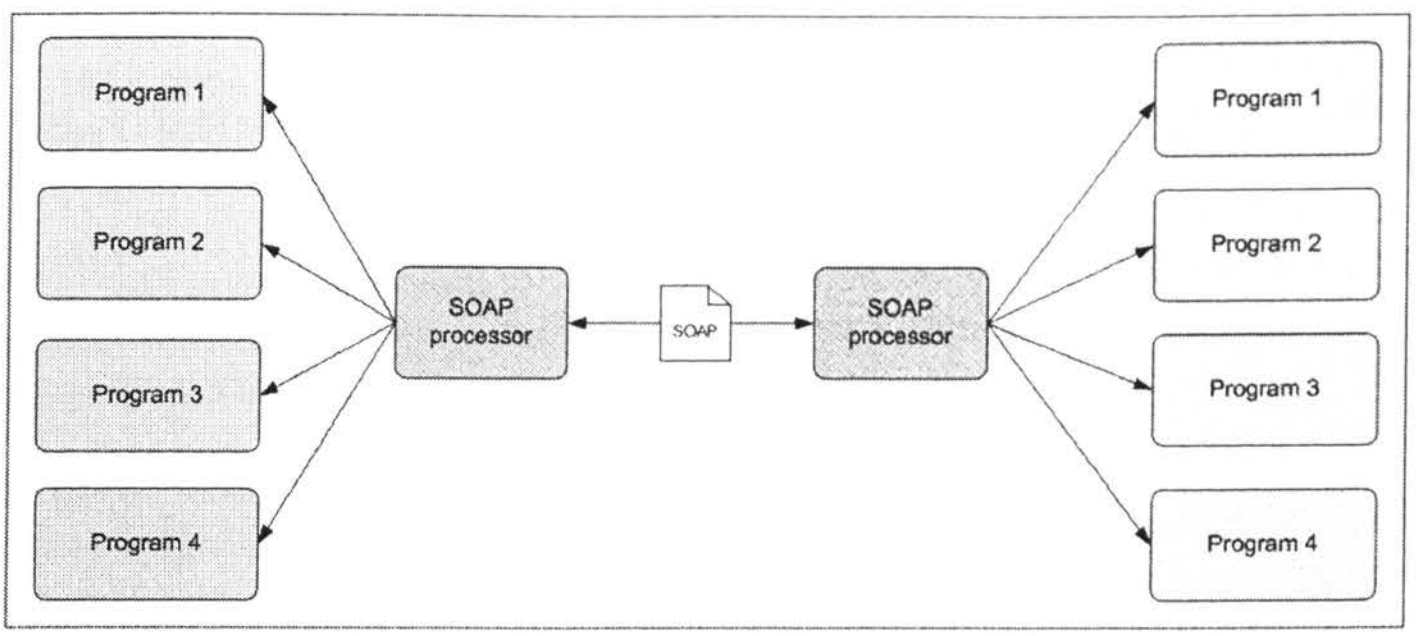
- Bezpečnosť
- Zaručené doručovanie správ
- Transakčnosť

Pri prepojení takých systémov pomocou SOAP, ktoré sa bez týchto vlastností nemôžu zaobísť, je nutné použitie rôznych rozšírení, ktoré tieto vlastnosti zaručujú.

SOAP je v podstate jednosmerný komunikačný model, ktorý slúži na prenos správy od odosielateľa k príjemcovi. SOAP podporuje aj prenášanie správ za pomoci prostredníkov (intermediares), ktorí môžu správu čiastočne spracovať a predať ďalej.

SOAP definuje aj mechanizmus ako je možné jednosmernú komunikáciu podporovanú SOAP-om adaptovať na dvojsmernú komunikáciu Request / Reply používanú pre

procedurálne orientovanej výmene správ, alebo ako pomocou SOAP prenášať kompletné dokumenty používané pri dokumentovo orientovanej výmene správ.

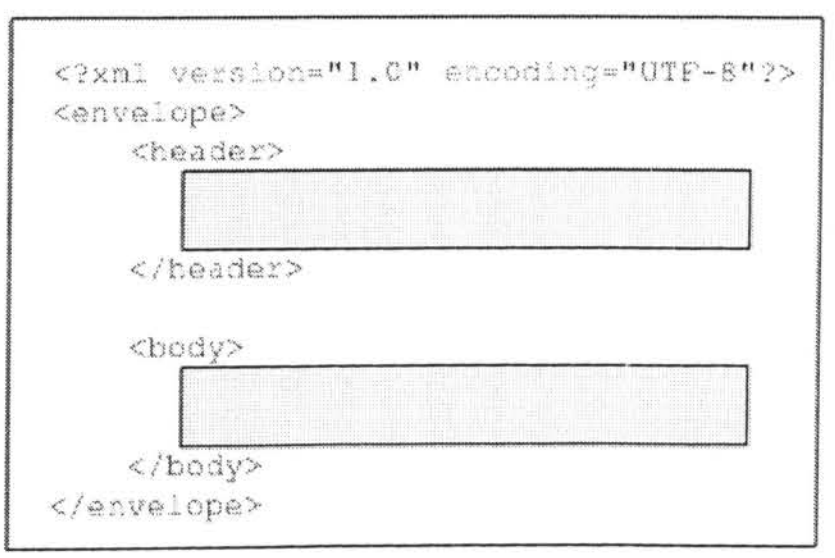


Obrázok 12 : Použitie SOAP-u

SOAP bol navrhnutý ako abstraktný, nezávislý komunikačný protokol vhodný pre premostenie alebo spojenie dvoch alebo viacerých komunikujúcich strán. Prepojené systémy môžu byť postavené na rôznej kombinácii hardwaru alebo softwaru, ktorý podporuje komunikáciu cez Internet, ako napríklad .NET alebo J2EE.

Samotná SOAP správa sa skladá z niekoľkých hlavných častí:

- Envelope – ohraničuje samotnú správu
- Header – Obsahuje metadáta používané pri spracovávaní správami komunikujúcimi stranami alebo prostredníkmi
- Body – samotné telo správy obsahujúce dáta
- Attachment – obsahuje jeden alebo viacej dokumentov priložených ku správe
- RPC interaction – definuje ako modelovať RPC volanie pomocou SOAP
- Encoding – definuje spôsob reprezentácie jednoduchých i komplexných dát prenášaných v tele správy. [20]



Obrázok 13 : Formát SOAP dokumentu

2.14. UDDI – Universal Distribution, Discovery and Interoperability

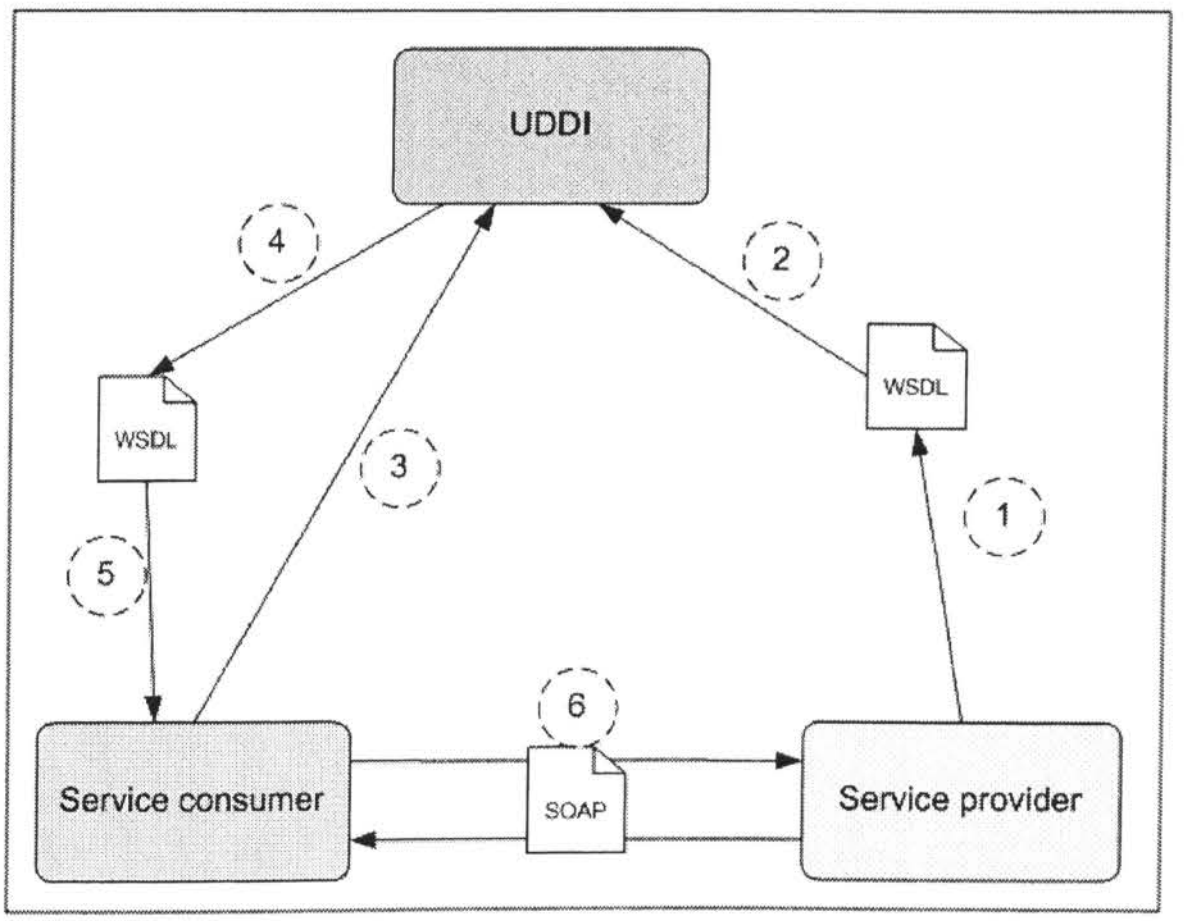
Podľa zastrešujúcej organizácie OASIS [21] je UDDI významný protokol pre vytváranie webových služieb. UDDI slúži na vytvorenie štandardnej platformy, ktorá umožňuje spoločnostiam a aplikáciám jednoducho, rýchlo a dynamicky vyhľadávať a používať webové služby a takisto jednoducho svoje webové služby publikovať a dosiahnuť tak situáciu, že ich služby môžu byť rovnako jednoducho, rýchlo a dynamicky používané inými spoločnosťami.

UDDI bol vytvorený nezávislým konzorciom softwarových spoločností, založeným spoločnosťami Microsoft, IBM a Ariba. Účelom konzorcia bolo vytvorenie štandardu pre popis, registráciu a vyhľadávanie webových služieb.

UDDI je konceptuálne podobný službe „Yellow pages". V „Yellow pages" spoločnosti registrujú svoje údaje ako telefónne číslo, fax, email a ďalšie kontaktné údaje. Pri registrácii sú tieto spoločnosti zaraďované do kategórií podľa zemepisnej polohy, oblasti podnikania, charakteru poskytovaných služieb. Iné spoločnosti následne môžu taktiež za pomoci služby „Yellow pages" v tomto registru vyhľadávať a zisťovať kontaktné údaje spoločnosti, ktoré ponúkajú služby, o ktoré majú záujem.

UDDI funguje obdobne ako „Yellow pages". Je možné do neho registrovať webové služby, ktoré je možné následne vyhľadávať a následne vďaka vráteným údajom o službe aj volať a komunikovať s nájdenými službami.

Pre komunikáciu s UDDI registrom sa používa SOAP a vďaka poskytnutému rozhraniu je možné za pomoci SOAP volaní volať funkcie pre pridávanie alebo vyhľadávanie služieb v rámci UDDI registra.



Obrázok 14 : Použitie UDDI

Typicky sa proces registrovania webovej služby do UDDI registra skladá z krokov:

1. Poskytovateľ služby musí najprv vygenerovať WSDL súbor popisujúci webovú službu, ktorú poskytuje.
2. Tento WSDL súbor sa registruje pomocou funkcie UDDI v registre. Po tomto vložení obsahuje UDDI register kontaktné informácie o ponúkanej službe vrátane WSDL súboru popisujúceho službu.
3. Potenciálny konzument služby vyhľadáva pomocou funkcie UDDI službu, ktorá vyhovuje jeho požiadavkám.
4. UDDI register vracia požadovaný WSDL súbor popisujúci službu.
5. Klient generuje odpovedajúce volanie za pomoci WSDL súboru.
6. Klient volá určitú funkciu služby za pomoci protokolu SOAP a poskytovateľ služby odpovedá taktiež pomocou správy vo formáte SOAP.

UDDI ako centrálny register nie je obmedzený na popisy služieb vo formáte WSDL. Popisy registrovaných služieb môžu byť v ľubovoľnom XML formáte ktorý popisuje službu. [22]

2.15. Ďalšie technológie webových služieb

Základné technológie webových služieb ako sú SOAP, WSDL a UDDI sú vhodné na premostenie dvoch technologických domén rôznych dodávateľov systémov. Pre rozumné použitie v rámci podniku, alebo napríklad pre výmenu citlivých dát medzi dvoma obchodnými partnermi je ale potrebná ďalšia množina noriem, ktoré vhodne rozšíria základné normy webových služieb o ďalšie dôležité vlastnosti a kvalitu služieb.

Proces obohatenia webových služieb o doplňujúce rozširujúce normy sa podobá úsiliu skupiny Object Management Group (OMG) pri vývoji Common Object Request Broker Architecture (CORBA). OMG definovalo komplexnú softwarovú architektúru podporujúcu transakčnosť, asynchrónne posielanie správ, bezpečnosť, zotavenie z chýb, atď. Podobné úsilie vyvíja konzorcium W3C ohľadom webových služieb.

Názov oblasti	Norma	Typ oblasti
Distribuovaný manažment	WSDM, WS-Management	Manažment
Súčinnosť	WS-Provisioning	
Bezpečnosť	WS-Security	Bezpečnosť
Bezpečnostná politika	WS-SecurityPolicy	
Bezpečná výmena dát	WS-SecureConversation	
Dôveryhodné posielanie správ	WS-Trust	
Federalizovaná správa identít	WS-Federation	
Portál	WSRP	Portál
Transakčnosť	WS-Transactions, WS-Coordination, WS-CAF	Transakčnosť a podpora procesov
Podpora procesov	BPEL4WS, WS-CDL	
Oprácie so správami	WS-Eventing, WS-Notification	Messaging
Zoskupovanie správ	WS-Enumeration, WS-Transfer	
Smerovanie správ	WS-Addressing, WS-MessageDelivery	
Spoľahlivosť správ	WS-ReliableMessaging, WS-Reliability	
Zaobaľovanie správ	SOAP, MTOM	
Publikácia a vyhľadávanie	UDDI, WSIL	Metadata
Pravidlá	WS-Policy	
Popis služieb	WSDL	
Získavanie metadát	WS-MetadataExchange	

Obrázok 15 : Rozširujúce technológie webových služieb

Proces obohatenia webových služieb o doplňujúce normy sťažuje fakt, že vedúce spoločnosti ktoré sa podieľali na vytvorení noriem SOAP, WSDL a UDDI sa už nevedia zhodnúť na jednotlivých rozširujúcich normách a každá presadzuje svoje vízie budúcnosti.

Rozšijúce normy by sa dali rozdeliť do nasledujúcich základných oblastí:

- Manažment
- Bezpečnosť
- Portál
- Transakčnosť
- Obsluha správ
- Správa procesov
- Metadáta

2.15.1. *Manažment*

Správu podnikových zdrojov a ich následnú integráciu podporujú nasledujúce doporučenia:

- WSDM – Web Services Distributed Management, ktorý podporuje správu zdrojov pomocou webových služieb, ako aj správu webových služieb ako určitého typu zdroja. [23]
- WS-Management – konkurenčná špecifikácia k WSDM, pokrývajúca len časť problému, ktorý pokrýva WSDM. [24]
- WS-Provisioning – norma popisujúca API schéma nutné pre zaistenie súčinnosti medzi systémami spravujúcimi zdroje pomocou webových služieb. [25]

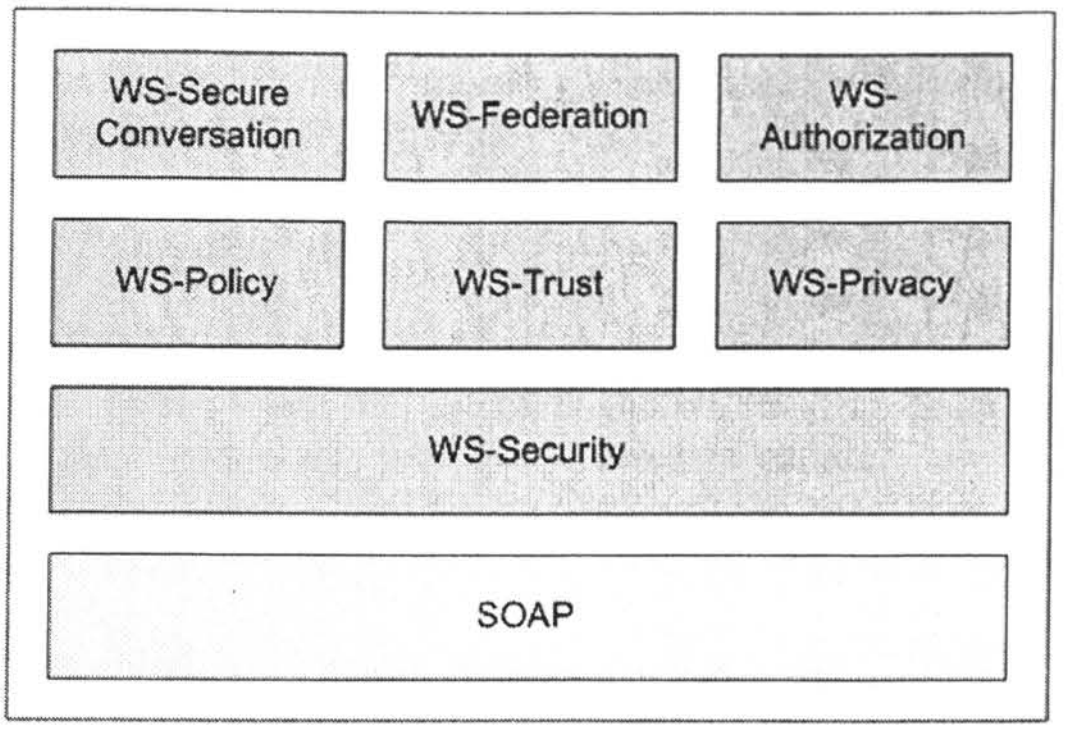
2.15.2. *Bezpečnosť*

Jednou z najdôležitejších rozširujúcich noriem sú normy ohľadom bezpečnosti webových služieb. Bezpečnosť je dôležitá pre zaistenie dôvernosti a integrity webových služieb. Nie je dovolené, aby niekto iný ako určený príjemca správ videl obsah dôverných správ a mohol s nimi svojvoľne manipulovať. Bezpečnostné normy sú dôležité aj pre zaistenie prístupu k webovým službám, aby webové služby mohol využívať len ten, pre koho sú určené.

Špecifikácie k bezpečnosti webových služieb:

- WS-Security – popisuje rozšírenie normy SOAP o ďalšie úrovne ochrany pomocou integrity, diskrétnosti a overovania správ. Tieto mechanizmy umožňujú použitie rôznych bezpečnostných modelov a šifrovacích technológií ako sú napríklad podpisy alebo bezpečnostné žetóny (tokeny) ako X.509 certifikáty alebo Kerberos tikety. [26]
- WS-SecurityPolicy – definuje množinu bezpečnostných vyhlásení týkajúcich sa noriem WS-Security, WS-Trust, a WS-SecureConversation. Popisujú ako majú byť prenášané správy zabezpečené, aby bola zároveň zaručená kompatibilita a súčinnosť medzi jednotlivými účastníkmi komunikácie. [27]
- WS-Trust – definuje rozšírenie normy WS-Security pre popis ďalších rozšírení výmeny bezpečnostných žetónov, ktoré slúži na overenie identity komunikujúcich strán navzájom pre zaistenie dôvery medzi stranami. [28]
- WS-SecureConversation – rozšírenie WS-Security a WS-Trust, ktoré popisuje ako spravovať a autentifikovať výmenu správ medzi komunikujúcimi stranami vrátane výmeny bezpečnostného kontextu a vytvorenia kľúčov pre spojenie, v rámci ktorého prebieha konverzácia (výmena viacerých správ). [29]
- WS-Federation – popisuje ako spravovať dôverné väzby (identity, bezpečnostné požiadavky) v heterogénnom prostredí rôznych bezpečnostných domén medzi webovými službami. [30]
- WS-Authorization – popisuje správu autorizačných dát a autorizačných postupov webovými službami. [31]
- WS-Privacy – popisuje ako webové služby definujú ochranu súkromia. Následne konzumenti týchto služieb musia tieto definície rešpektovať. [32]

Vzťah medzi hlavnými špecifikáciami ohľadom bezpečnosti webových služieb popisuje nasledujúci obrázok.



Obrázok 16 : Špecifikácie bezpečnosti webových služieb

2.15.3. Portál

Pre potreby priameho integrovania webových služieb do užívateľského rozhrania - preto vznikla norma WSRP.

- WSRP (Web Services for Remote Portlets) je špecifikácia, ktorá definuje volanie webových služieb priamo z portálov. WSRP taktiež umožňuje u distribuovaných portálov zdieľať portlety ako virtuálne webové služby využiteľné v iných portáloch. [33]

2.15.4. Podpora procesov

Process flow (správa procesov) je kritická pre automatizovanie business procesov v rámci podniku. Správa procesov sa často nazýva orchestrácia, lebo popisuje vzťah medzi sériou interakcií nutných pre vykonanie daného cieľu, ako sú napríklad prevedenie objednávky, spracovanie rezervácie alebo splnenie určitého plánu. Procesy sú modelované ako postupnosť krokov definovaných pre daný business proces. Postupnosť krokov tvorí združenie funkcií, pre ktoré je možné definovať webovú službu.

Základné normy pre podporu správy procesov sú:

- XLANG – rozšírenie WSDL vytvorené spoločnosťou Microsoft pre podporu správy procesov v rámci webových služieb. Umožňuje spájať jednotlivé služby do reťazca dlhodobých transakcií. [34]
- WSFL – XML jazyk definujúci popis možného zostavovania jednotlivých webových služieb do zložitých sekvencií. Definuje poradie volania služieb i dátového toku medzi jednotlivými volanými službami. Bol vytvorený spoločnosťou IBM. WSFL je vlastne konkurenčnou normou k norme XLANG. [35]
- BPEL4WS – jazyk pre definovanie workflow modelu obsahujúceho transakcie skladajúce sa z definovaných sekvencií webových služieb. Túto normu podporujú mimo iných aj spoločnosti IBM, BEA a Microsoft. BPEL4WS vznikol z XLANG a WSFL a nahrádza ich. [36]

- WS-CDL (Choreography Description Language) – jazyk pre popis vytvárania spolupráce medzi koncovými komunikujúcimi bodmi nezávisle na platforme alebo programovacom modelu. [37]

2.15.5. Transakčnosť

Vo svete automatizovaných business operácií hrajú transakcie dôležitú úlohu. Zaisťujú korektnosť prevedenia série súvisiacich operácií nad dátami, a vďaka nim sa zložité viac krokové operácie navonok javia ako atomické operácie nad dátami.

Tradičné transakčné protokoly nie sú v rámci webových služieb podporované.

Riešením pre tento problém je norma od pracovnej skupiny OASIS nazvaná Bussiness Transaction Protocol (BTP) , ktorá zaisťuje, že výsledky volania viacerých webových služieb sú korektne propagované a zdieľané. [38]

Medzi ďalšie normy pre podporu transakčnosti webových služieb patria :

- WS-Transaction – definuje mechanizmus pre podporu transakčnej súčinnosti medzi rôznymi doménami webových služieb a prácu s transakciami nad webovými službami. [39]
- WS-Coordination – poskytuje protokol pre koordinovanie akcií distribuovaných aplikácií. [40]
- WS-CAF (Composite Application Framework) – jazyk pre štandardnú podporu prenášania kontextu, koordináciu a správu transakcií. Skladá sa z troch špecifikácií:
  - WS-CTX (Context) - jednoduchý framework pre správu kontextu
  - WS-CF (Coordination Framework) – definuje centrálneho koordinátora, ktorý spravuje kontextové informácie ako aj správu propagovania úspešnosti volania jednotlivých služieb
  - WS-TXM (Transaction Management) – definuje tri transakčné protokoly, ktoré môžu byť použité v koordinačnom frameworku pre zaistenie transakčnosti volaných sekvencií služieb.[41]

2.15.6. Obsluha správ

Messaging protocols (protokoly pre správu správ) zaisťujú správne fungovanie rôznych komunikačných scenárov používaných pri výmene správ, ako sú:

- Asynchrónna jednosmerná komunikácia
- Request / Response
- Broadcast
- Publish / Subscribe

Ďalšie webové služby môžu závisieť na určitej kvalite služieb ako sú:

- Spoľahlivé (zaručené) doručovanie správ
- Propagovanie transakčného alebo bezpečnostného kontextu
- Správne smerovanie doručovania správy cez definovaných prostredníkov

Normy riešiace tieto problémy sú:

- WS-Routing – protokol pre definovanie doporučenej cesty posielanej správy podporujúci široké spektrum transportných technológií [42]
- WS-Referral – protokol pre možnú dynamickú konfiguráciu SOAP routerov pomocou hlavičky SOAP správy [43]
- Blocks Extensible Exchange Protocol (BEEP) – framework pre vytvorenie aplikačného protokolu pre asynchrónnu komunikáciu pomocou správ zvyčajne založených na XML [44]
- HTTPR – HTTP Reliable, protokol zaručujúci spoľahlivé doručovanie HTTP paketov medzi komunikujúcimi stranami. Rieši množstvo problémov nespoľahlivého HTTP protokolu. [45]
- WS-Eventing – definuje základnú množinu operácií, ktoré môžu webové služby používať za účelom ohlásenia (notifikácie) určitej udalosti. [46]
- WS-Notification – norma pre definovanie Publish / Subscribe scenára posielania správ pre webové služby. [47]
- WS-Enumeration – norma definujúca XML konštrukciu pre získanie dát, ktoré majú tvar kolekcie od webových služieb. [48]
- WS-Transfer – definuje nad webovými službami množinu jednoduchých operácií (Get, Post, Put, a Delete) a definuje jednotnú možnosť ako ich volať. [49]
- WS-Addressing – definuje mechanizmus identifikovania koncových bodov komunikácie nezávisle na transportnej vrstve, vďaka ktorému je možné jednoduché adresovanie správ. [50]
- WS-MessageDelivery – táto norma rieši ten istý problém ako norma WS-Addressing, ale je integrovaný do WSDL a používa jeho tagy, zatiaľ čo WS-Adressing vytvára externé „referenčné vlastnosti". [51]
- WS-ReliableMessaging – definuje doručovací protokol nezávislý na transportnej vrstve pre spoľahlivé doručenie správ medzi komunikujúcimi stranami. Definuje takisto SOAP binding pre využitie v rámci webových služieb. [52]
- WS-Reliability – rozšírenie normy SOAP, ktoré dodáva podporu pre spoľahlivé doručovanie správ, elimináciu duplicitných správ, radenie doručovania správ. [53]
- MTOM (Message Transmission Optimization Mechanism) – popisuje mechanizmus optimalizácie správ medzi dvoma jednotlivými bodmi komunikačnej cesty. Optimalizácia sa definuje vždy medzi dvoma SOAP uzlami (odosielateľ, príjemca, prostredníci) a je založená na selektívnom kódovaní jednotlivých častí SOAP správy. [54]

2.15.7. Metadáta

Poslednou skupinou noriem sú normy zaisťujúce pridanú hodnotu v oblasti práce s metadátami – dáta o dátach.

- WSIL (Web Service Inspection Language) – norma známa aj pod názvom WS-Inspection, XML formát pre podporu dynamického vyhľadávania dostupných webových služieb v rámci vzdialeného priestoru. Táto norma umožňuje aplikáciám vyhľadávanie služieb, o ktoré majú záujem. [55]
- WS-Policy – popisuje flexibilnú gramatiku pre vyjadrenie požiadaviek, možností a základných charakteristík jednotlivých webových služieb. Tieto vlastnosti sú

vyjadrené pomocou nariadení. Následne môžu užívatelia i poskytovatelia webových služieb definovať svoje nároky na minimálne požiadavky. [56]

- WS-MetadataExchange – norma, ktorá zavádza mechanizmus metadáta - orientovanej výmeny správ za pomoci WSDL alebo WS-Policy. Norma deklaruje snahu podporovať aj budúce formáty metadát. [57]

2.16. ESB (Enterprise Service Bus)

Z viacerých možných definícií tohto často sa vyskytujúceho pojmu vyplýva, že ESB by sa dala definovať ako určitý architektonický vzor pre middleware, ktorý slúži na prepojenie a riadené zdieľanie podnikových zdrojov (služieb, aplikácií, dát) bez ohľadu na technologickú rôznorodosť týchto zdrojov.

Pre vytvorenie servisne orientovanej architektúry je nutne sa držať princípov SOA. Pre vytvorenie služieb je nutné vytvoriť rozhrania pre existujúce alebo nové služby buď priamo, alebo za použitia adaptérov. Vytvorenie infraštruktúry na najnižšej úrovni znamená umožniť prenos a smerovanie správ reprezentujúcich volania jednotlivých služieb. Taktiež je dôležité umožniť zmenu implementácie služby bez dôsledkov na klientov danej služby. Toto je podmienené nielen vytvorením servisne orientovaných služieb, ale aj určitou podporou zo strany infraštruktúry volať služby nezávisle na skutočnej polohe služby a povahe komunikačného protokolu.

ESB poskytuje infraštruktúru pre komunikáciu, posielanie správ a správu udalostí, pomocou ktorej je možné vytvoriť servisne orientovanú architektúru. Ďalej ESB poskytuje nástroje pre hlavné integračné vzory.

Rôzna literatúra popisuje rôzne vlastnosti ESB. Niektoré z týchto vlastností tvoria základ ESB ale niektoré na druhú stranu tvoria rozšírenú funkčnosť ESB. Zoznam vlastností poskytovaných rôznymi implementáciami ESB:

- Komunikácia
  - Smerovanie správ (routing)
  - Adresovanie správ (addressing)
  - Podpora komunikačných technológií a štandardov
  - Publish / subscribe
  - Request / response
- Prepojenie služieb
  - Definícia rozhrania služby
  - Podpora pre zámenu implementácie služby
  - Modely výmeny správ
  - Vyhľadávanie a zverejňovanie služieb
- Integrácia
  - Databáza
  - Adaptéry
  - Skladanie služieb
  - Mapovanie služieb
  - Transformácia protokolov
  - Podpora aplikačných serverov
  - Podpora rôznych jazykových rozhraní pre volanie služieb
- QoS

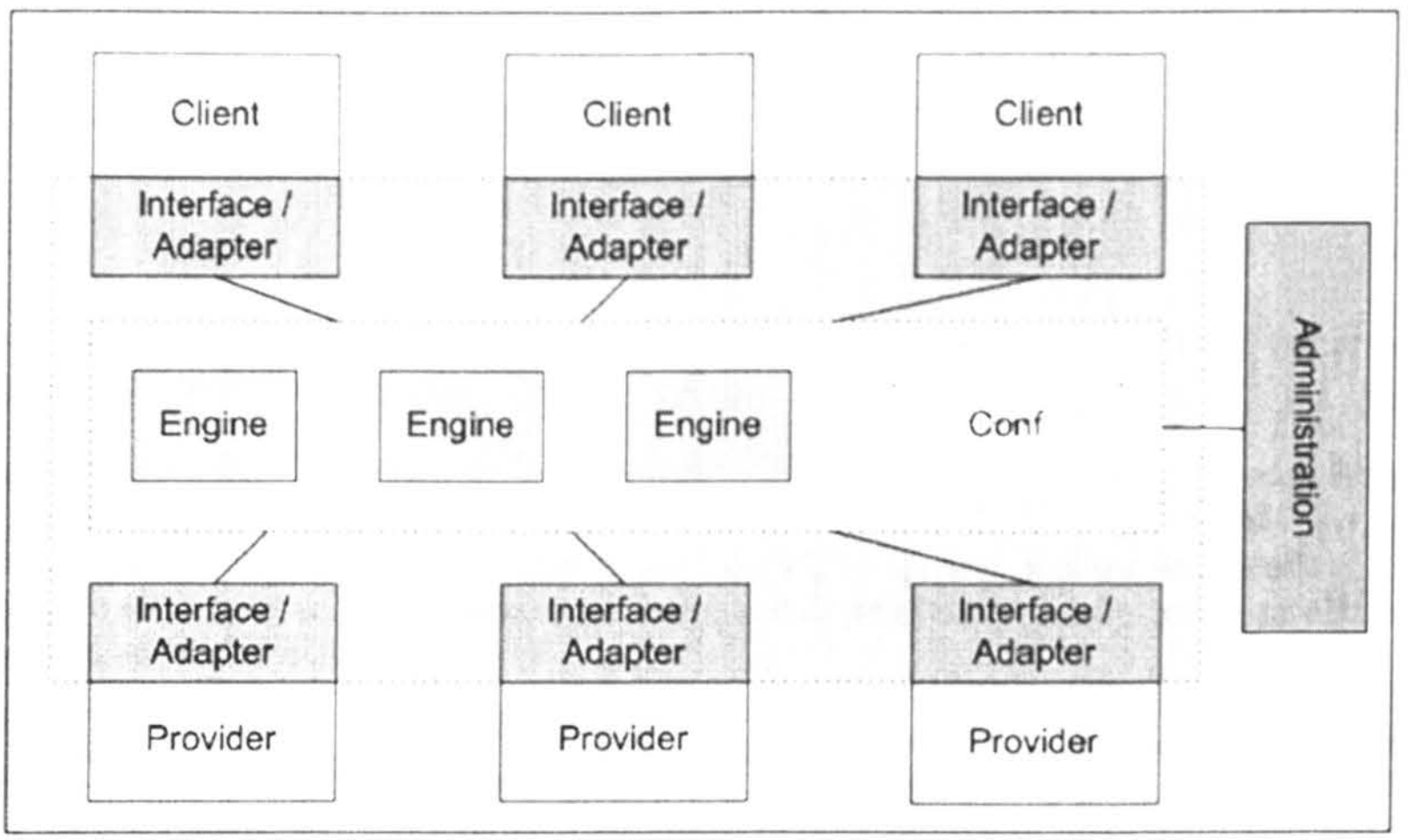
    - Transakcie
    - Podpora zaručeného posielania správ
- Bezpečnosť
    - Autentifikácia
    - Autorizácia
    - Dôvernosť
    - Podpora štandardov bezpečnosti
- Úroveň služieb
    - Výkonnosť
    - Priepustnosť
    - Dostupnosť
- Spracovanie správ
    - Validácia
    - Transformácia správ
    - Obohacovanie správ
- Manažment
    - Registrácia služieb
    - Loging, monitoring
    - Vyhľadávanie služieb
- Modelovanie
- Inteligencia infraštruktúry

Z týchto ponúkaných vlastností je ale len určitá množina obsiahnutá medzi základnými vlastnosťami nutnými k dosiahnutiu ESB. Tieto vlastnosti sú :

- Komunikácia
    - Smerovanie správ (routing)
    - Adresovanie správ (addressing)
    - Administratívna podpora smerovania a adresovania
    - Aspoň jeden spôsob výmeny správ (Publish / subscribe, Request / response ..)
    - Aspoň jeden transportný protokol
- Prepojenie služieb
    - Otvorený model pre výmenu správ, ktorý oddeľuje smerovanie správ od samotnej aplikácie s podporou zámeny implementácie služby
- Integrácia
    - Podpora integrácie služieb (napr. webové služby, adaptéry, Java 2 Connectors)

Tieto základné vlastnosti splňujú najčastejšiu definíciu ESB:

- ESB je logický architektonický komponent poskytujúci infraštruktúru rešpektujúcu princípy SOA
- Požaduje použitie rozhraní nezávislých na implementácii, komunikačné protokoly nezávislé na umiestnení služby, prenosnosť a znovupoužiteľné služby s rozhraním so slabou väzbou.
- ESB je implementovaná ako distribuovaná heterogénna infraštruktúra
- ESB poskytuje nástroje pre správu infraštruktúry [58]



Obrázok 17 : ESB

3. TIF

3.1. Popis

Trask Integration Framework (TIF), ako už názov napovedá je integračný framework spoločnosti Trask solutions s.r.o.

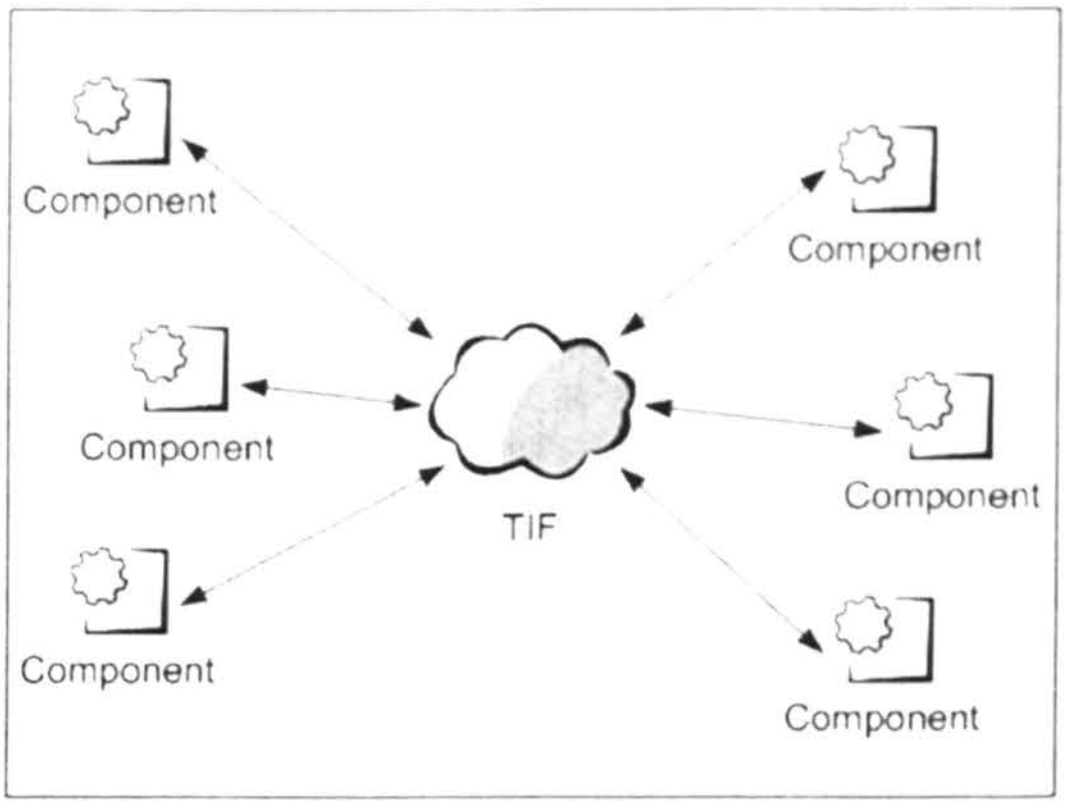
Trask Integration Framework by sa dal primárne označiť ako implementácia architektonického vzoru označovaného ako Enterprise Service Bus (ESB), vhodný pre integráciu interných autonómnych vnútropodnikových komponent v rámci konfederácie. TIF prepojuje jednotlivé servisne orientované komponenty (aplikácie) konfederácie ponúkajúce a konzumujúce služby.

TIF taktiež umožňuje flexibilné riešenie integračných scenárov ako sú Event Driven Architecture (EDA), Exctract, Transform and Load (ETL) a B2B integrácia.

Rozhrania jednotlivých komponentov poskytujúcich služby tvoria business orientované funkcie implementujúce procesy ponúkané komponentmi. Rozhranie komponentov je verejne známe v rámci konfederácie, zatiaľ čo implementácia komponentu môže mať charakter čiernej skrinky. Komponenty tvoria koncové body virtuálnej siete (peer), ktorá vznikne prepojením za pomoci TIF-u.

Prepojením jednotlivých komponentov konfederácie do virtuálnej P2P siete konzumentov a poskytovateľov business orientovaných služieb pomocou TIF-u vzniká architektúra orientovaná na služby.

TIF ponúka možnosť jednoduchého a rýchleho nasadenia bez nutnosti customizovania produktu u zákazníka. Nasadenie produktu spočíva v inštalácii a konfigurácii jednotlivých komponentov TIF-u. Táto vlastnosť je zaručená pomocou širokej podpory rozličných komunikačných protokolov pre integráciu komponentov.



Obrázok 18 : Príklad riešenia pomocou TIF-u

3.2. Pojmy

V texte sa používajú pojmy ako frontend a backend.

- Frontend je iný názov pre konzumenta služby v architektúre orientovanej na služby.
- Backendom je na druhú stranu označovaný poskytovateľ.

3.3. História

3.3.1. Na začiatku bol len jeden projekt

Prvá verzia produktu Trask Integration Framework, ktorý sa dnes už dá považovať za ESB, vznikla v priebehu implementácie nového systému pre obsluhu klientov v spoločnosti GE Capital Bank, a.s. (dnes GE Money Bank, a.s.) už v priebehu rokov 2002 až 2004. V priebehu analýzy bolo identifikovaných dvanásť veľmi rôznorodých aplikácií, s ktorými sa systém pre obsluhu klientov musel integrovať. Integrované systémy boli prevádzkované na niekoľkých rôznorodých operačných systémoch, využívali rôzne dátové zdroje a boli postavené na odlišných programátorských technológiách. Vzhľadom k tejto rôznorodosti bolo rozhodnuté o implementácii distribuovanej integračnej infraštruktúry založenej na princípoch SOA, ktoré by minimalizovali dopady na integrované aplikácie a do budúcna by umožňovali naďalej rozširovať integračnú infraštruktúru o ďalšie riešenia a aplikácie.

Hlavné požiadavky kladené na pripravované riešenie boli :

- **Znovupoužiteľnosť** – Nielen na úrovni jednotlivých služieb ale aj na úrovni celej integračnej infraštruktúry.
- **Granularita a modularita** – Cieľom bolo pripraviť moduly a služby pokrývajúce definované funkčnosti.
- **Interoperabilita** – Potreba vzájomnej komunikácie medzi aplikáciami na rôznych platformách, s rôznymi dátovými zdrojmi, pripravených pomocou rôznych technológií rôznymi dodávateľmi.

- **Voľná väzba** – Riešenie malo zachovať maximálnu voľnosť medzi komunikujúcimi aplikáciami tak, aby napr. zmeny v aplikáciách alebo jednotlivých službách nemali dopad za hranicu integračného rozhrania, alebo aby boli tieto dopady minimalizované.
- **Štandardy** – Bolo požadované štandardizovať integračnú infraštruktúru v zmysle spätnej kompatibility verzií a možnosti ďalšieho rozvoja bez nutnosti veľkých alebo častých zmien. (pozn.: paradoxne sa ukázalo, že pre naplnenie tohto cieľu je výhodnejšie sa vyvarovať používania priemyslových „štandardov" spojených napr. s webovými službami. Tieto štandardy sa menia a rozširujú spôsobom, že je veľmi obtiažne zaistiť konzistenciu, spätnú kompatibilitu, ale i kompatibilitu medzi rôznymi časťami riešenia).
- **Centrálna databáza služieb** – Aby bolo možné integračné služby efektívne využívať z rady vývojových nástrojov v rámci projektu, musela vzniknúť centrálna databáza služieb, ktorá obsahuje nielen informácie o jednotlivých službách, ale aj informácie o číselníkoch a ich mapovaní na hodnoty v konkrétnych aplikáciách.
- **Bezstavovosť** – Táto požiadavka vyplynula v priebehu riešenia a bola implikovaná najmä potrebou zaistenia vysokej dostupnosti a rozkladania záťaže – bezstavovosť je takmer nevyhnutným predpokladom pre implementáciu takéhoto riešenia.
- **Správa a dohľad** – Riešenie muselo poskytnúť nástroje pre efektívny dohľad a správu celej integračnej infraštruktúry.

3.3.2. Webové služby vs. messaging

Ako komunikačné technológie boli posudzované varianty využitia webových služieb nad http (konkrétne s nasadením IBM WebSphere Application Serveru) a naproti tomu Message Oriented Middleware (MOM) (konkrétne IBM WebSphere MQ).

Behom úvodného nasadenia bolo rozhodnuté orientovať sa predovšetkým na WebSphere MQ. Hlavným dôvodom je, že WebSphere MQ je produkt priamo určený pre integráciu rôznorodých aplikácií v heterogénnom prostredí, naproti tomu webové služby v lepšom prípade predstavujú štandard pre vlastné technologické rozhranie, ale na integračnom týme stále ostáva riešenie rady rozsiahlych problémov, ktoré webové služby neriešia.

Najväčšie obavy spojené s nasadením WebSphere MQ boli spojené s možnou časovou réžiou (t.j. s dlhšou dobou na prenos správ), ktorá je z princípu u produktov kategórie MOM vyššia než u priamej synchrónnej komunikácii. Boli preto vykonané záťažové testy, ktoré ale tieto obavy vyvrátili.

3.3.3. Prístup k budovaniu integračnej infraštruktúry

Okrem voľby WebSphere MQ ako integračnej technológie, ktorá významne ovplyvnila podobu TIF, bola taktiež riešená otázka, akým spôsobom vybudovať vlastnú integračnú infraštruktúru. V priebehu projektu boli zvažované tieto dve hlavné varianty :

- Definovať pravidlá pre dodávateľov aplikácií – na úrovni integračného týmu definovať pravidlá pre prácu so správami MQ (formáty, spôsob volania, atď.) a ponechať ťažisko na riešiteľoch zodpovedných za jednotlivé integrované aplikácie. Toto je cesta, ktorou sa veľmi často integračné projekty uberajú.
- Implementovať integračnú infraštruktúru v rámci projektového týmu – pripraviť v rámci projektu sadu modulov, ktoré pokrývajú všeobecné požiadavky kladené na

integračnú infraštruktúru (zahrňuje napr. jednotný spôsob logovania, auditu, konzistentné zabezpečenie komunikácie, validáciu správ a dátových typov, jednotný spôsob ošetrenia stavov, konzistentný spôsob správy integračných modulov atď.).

Rozhodnutie padlo v prospech druhej varianty implementovať integračnú infraštruktúru v rámci projektového týmu. Dôvody pre tento výber boli nasledovné :

- Pestrosť aplikácií a ich dodávateľov – významne by skomplikovalo situáciu v prípade, že by boli len stanovené pravidlá a dodávatelia by garantovali ich splnenie. Medzi zrejmé riziká by patrili : nutnosť komunikácie a vysvetlenia veľkému počtu dodávateľov, nutnosť komplexných testov (chyby dodávateľov) a v neposlednej rade taktiež možnosť chýb v špecifikácii – každá taká chyba by potenciálne ovplyvnila všetkých dodávateľov.
- Časový faktor a ceny – ako je zrejmé z predchádzajúceho bodu, varianta, keď každý dodávateľ implementuje podľa definovaných pravidiel, znamenala významné dopady do harmonogramu projektu a taktiež do ceny dodávok. Najviac je zrejmé, že by všetci dodávatelia museli riešiť podobné implementačné zadanie a zbytočne by sa tak opakovane implementovali rovnaké mechanizmy, len rôznymi dodávateľmi a rôznym spôsobom.
- Znovupoužiteľnosť – raz pripravené integračné moduly je možné opakovane využiť pre ďalšie riešenia a aplikácie a budovať koncepčnú integračnú infraštruktúru organizácie.

3.3.4. Úspech projektu

Ukázalo sa ako výhodné, že celá integračná infraštruktúra bola vybudovaná v rámci projektu nového klientského systému – riešenie bolo pripravené rýchlo, dopady na projekt a dodávateľov boli minimalizované a je možné povedať, že už v priebehu jedného projektu sa náklady na vybudovanie integračnej infraštruktúry zákazníkovi zúročili.

Organizácia zákazníka naviac získala prvotriednu integračnú infraštruktúru na báze SOA, ktorá umožnila IT oddeleniam organizácie pracovať s oveľa väčšou flexibilitou a reagovať rýchlejšie na požiadavky businessu s dopadmi do viacerých systémov.

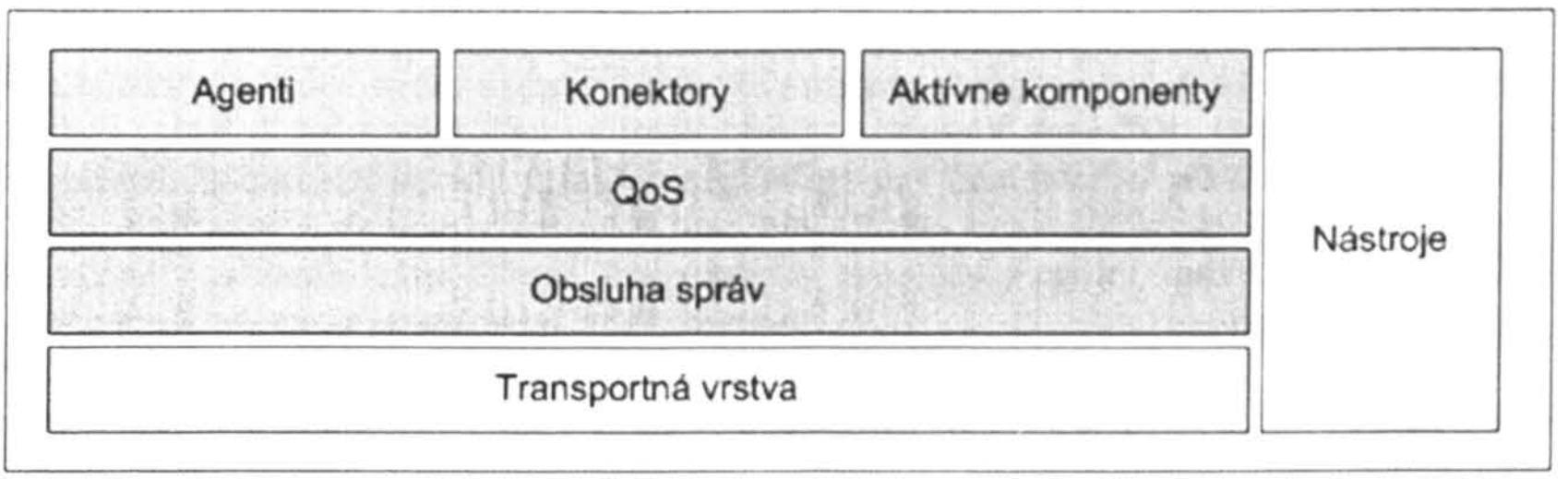
3.3.5. To ostatné je už história

Moduly dodané v rámci projektu boli od začiatku koncipované ako celkom obecné integračné nástroje použiteľné pre ľubovoľné integračné aktivity. Ich najväčšou výhodou je jednoduchosť a znovupoužiteľnosť s maximálnym dôrazom na efektivitu nasadenia a prínos pre zákazníka.

Vzniknuté moduly bolo spoločnosťou Trask solutions vzaté ako základ produktu Trask Integration Framework, ktorý je ďalej rozvíjaný a rozširovaný, používaný v rade ďalších riešení a u ďalších zákazníkov a predstavuje vynikajúci základ pre koncepčné budovanie integračnej infraštruktúry v skutočnom zmysle pojmu Enterprise Service Bus.

3.4. Architektúra

Architektúra TIF-u je znázornená na nasledujúcom obrázku.



Obrázok 19 : Zjednodušená architektúra TIF-u

- **Transportná vrstva** – vrstva zaručujúca prenášanie správ medzi jednotlivými bodmi P2P siete.
- **Obsluha správ** – vrstva zaobaľujúca jednotlivé požiadavky a ich odpovede do správ definovanej formy
- **QoS** – vrstva poskytujúca definovanú kvalitu služieb
- **Agenti** – typové adaptéry (brány) pre univerzálne volanie rozhrania rôznych typových backendov
- **Konektory** – jednoducho použiteľné knižnice pre integráciu TIF-u do frontendových aplikácií konzumujúcich služby
- **Aktívne komponenty** – množina aktívnych komponentov, ktorá rozširuje integračné možnosti frameworku
- **Nástroje** – množina užitočných nástrojov, ktoré zjednodušujú spravovanie a prácu so vzniknutou infraštruktúrou

3.4.1. Transportná vrstva

Transportná vrstva slúži na prenos správ vytváraných za účelom vzájomnej komunikácie medzi jednotlivými koncovými bodmi virtuálnej P2P siete. Koncové body tejto P2P siete sú tvorené konzumentmi alebo poskytovateľmi služieb. Úloha týchto koncových bodov je v rámci infraštruktúry ale rovnocenná.

Transportná vrstva je primárne tvorená produktom firmy IBM, IBM WebSphere MQ. Tento produkt bol vybraný s prihliadnutím na predchádzajúce skúsenosti s jeho používaním a po skúsenostiach s jeho nasadením u zákazníkov v iných projektoch.

TIF však nie je úzko spätý s touto implementáciou transportnej vrstvy. Je možná jednoduchá zmena architektúry a použitie iného MOM alebo inej technológie na prenos správ.

Pre TIF sú dôležité nasledujúce kľúčové vlastnosti poskytované MQ :

- Zaručené doručenie správ práve raz
- Asynchrónny prístup k správam

- Jednoduchý monitoring aktivity
- Široká podpora platforiem
- Robustnosť a spoľahlivosť
- Stabilita a jednoduchosť rozhrania

Z hľadiska spoľahlivosti transportnej vrstvy je zaručené doručenie správ dôležité. Dokonca by sa dalo povedať, že kritické. Transportná vrstva musí zaručiť prenášanie určitých správ medzi koncovými bodmi infraštruktúry, inak je celý integračný framework v praxi nepoužiteľný.

Zaručené posielanie správ okrem straty správy zamedzuje taktiež duplikovaniu správ pri znovu posielaní správ považovaných za stratené.

Doručenie správy je zaručené dokonca aj v prípade, že cieľový bod alebo niektorý prvok na ceste je dočasne nedostupný.

Prístup k jednotlivých správam však všeobecne nie je rovnaký. Zaručené doručovanie správ vyžadujú najmä správy reprezentujúce aktívne operácie meniace dáta v cieľových systémoch (napr. platobný príkaz). Na druhú stranu pasívne operácie (napr. zostatok na účte) nevyžadujú v takej miere nutnosť zaručeného doručovania správ, pretože nemôžu spôsobiť nekonzistenciu dát.

Asynchrónnosť správ je dôležitá pre zaručenie funkčnosti systému napr. v prípade, že cieľové systémy bežia v iných časových intervaloch než zdrojové systémy. Táto asynchrónnosť je zaručená pomocou asynchrónnych front, ktoré posielané správy podržia až do času, pokiaľ ich cieľový prvok z fronty nevyberie. Rýchlosť asynchrónneho doručovania správ je v prípade bežiaceho systému prakticky rovnaká ako rýchlosť synchrónneho doručovania správ. Napr. odosielanie správ pre datawarehouse realtime.

MQ poskytuje jednoduchú možnosť monitoringu aktivity transportnej vrstvy. Tento monitoring slúži na vyhodnocovanie vyťaženia jednotlivých bodov infraštruktúry. V prípade preťaženia je možné následne podniknúť nápravné kroky pre zaručenie plynulosti fungovania integračného riešenia.

Monitoring taktiež umožňuje sledovanie stavu jednotlivých koncových bodov infraštruktúry. Na základe sledovania zaplnenia fronty cieľového systému je napr. možné identifikovať časy, v ktorých systém výkonnostne nestačí odpovedať na požiadavky.

MQ je podporované na širokej palete platforiem od z/OS (mainframe), cez UNIX, LINUX, iSeries (AS/400) až po MS Windows. Táto vlastnosť poskytuje väčšie možnosti nasadenia integračného riešenia u širšieho okruhu potenciálnych zákazníkov.

Robustnosť a spoľahlivosť transportnej vrstvy patrí k základným predpokladom komerčného nasadenia.

Stabilita a jednoduchosť rozhrania pre prácu s transportnou vrstvou zaručuje dostatočne efektívne a zároveň jednoduché použitie tejto vrstvy v rámci integračného frameworku. Taktiež garantuje, že nebude nutné systém prepracovať pri zmene štandardov, čo je napr. problém webových služieb.

Ďalšími podstatnými vlastnosťami MQ sú :

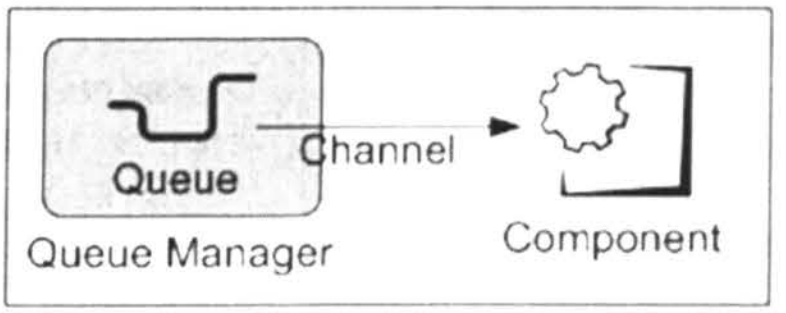
- Podpora bezpečnosti – MQ podporuje štandard SSL pre zabezpečenie prenosu správ na úrovni transportnej vrstvy. Tento bezpečnostný štandard zamedzuje neoprávneným osobám odchytávanie čitateľných citlivých dát.
- Dynamické distribuovanie záťaže – MQ podporuje pomocou clusteringu dynamické distribuovanie záťaže cez existujúce zdroje. Vďaka tejto vlastnosti je možné vytvoriť dostatočne robustnú infraštruktúru pre integračné riešenie.
- Podporné nástroje na správu transportnej vrstvy – nástroje pre vzdialenú správu transportnej vrstvy, jej administráciu a konfiguráciu. Umožňujú za behu optimalizovať infraštruktúru transportnej vrstvy a reagovať tak na požiadavky kladené na riešenie dynamicky.

Základnými stavebnými kameňmi MQ sú :

- Queue – Objekty uchovávajúce jednotlivé správy obsahujúce binárne dáta v rámci transportnej vrstvy.
- Queue Manager – Systémová služba poskytujúca logický kontajner pre Queue a zodpovedá za vymieňanie správ s inými Queue Managermi a komunikáciu s klientmi.
- Channel – Systémová služba pre zaistenie komunikácie medzi Queue Managermi navzájom aj s ich klientmi.

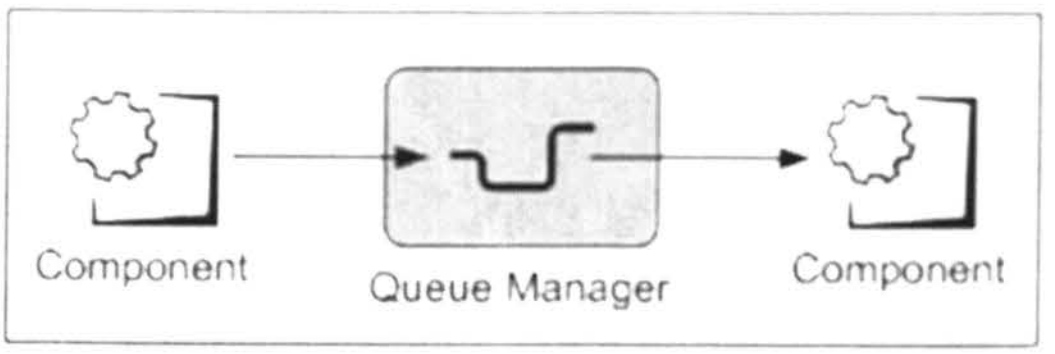
Každý aktívny komponent prijímajúci správy musí mať priradenú frontu cez ktorú prijíma správy od iných komponentov. Z tejto fronty môže odoberať správy len komponent, ktorému sú tieto správy určené. Pokiaľ by z jednej fronty odoberalo správy viac komponentov, mali by tieto komponenty vykonávať rovnakú činnosť, aby nedošlo k nedeterministickým chovaniam systému. Táto fronta reprezentuje koncový bod topológie.

Fronta komponentu je asynchrónna, udržiava prijaté správy až do ich vyzdvihnutia komponentom.



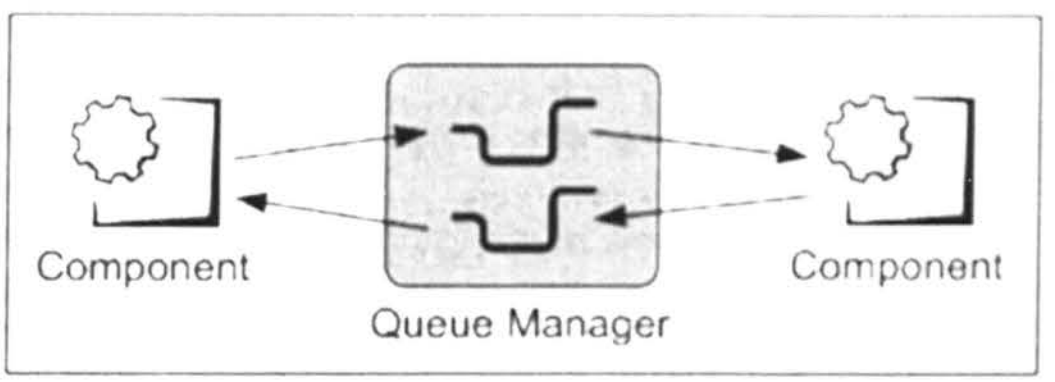
Obrázok 20 : Pripojenie komponentu k transportnej vrstve

Pre jednoduché zapojenie jedného odosielateľa stačí odchádzajúce správy odosielajúceho komponentu nasmerovať do fronty prijímajúceho komponentu. Takto vznikne prepojenie medzi odosielateľom a prijímateľom správy. Odosielajúci komponent môže generovať správy pre prijímajúci komponent, ktorý ich jednu po druhej postupne spracuje.



Obrázok 21 : Vytvorenie jednosmernej komunikácie

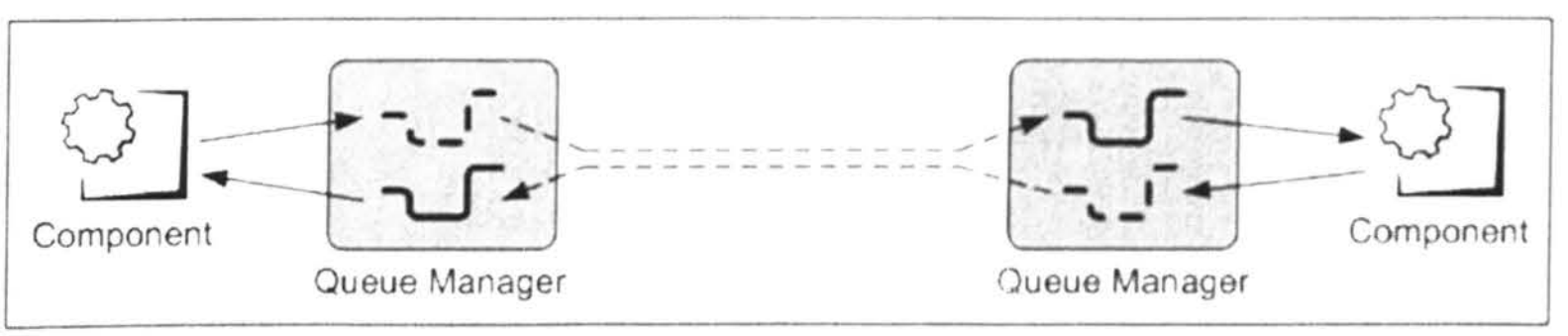
Pre umožnenie obojsmernej komunikácie medzi dvoma komponentmi je potrebná minimálne jedna fronta v každom smere prenosu správ. Táto požiadavka vyplýva z faktu že fronta uchováva správy putujúce len v jednom smere medzi komponentmi. Správy posielané cez fronty sú rovnocenné a ich rozlišovanie na požiadavku a odpoveď by v prípade použitia len jednej fronty bolo nemožné.



Obrázok 22 : Vytvorenie obojsmernej komunikácie

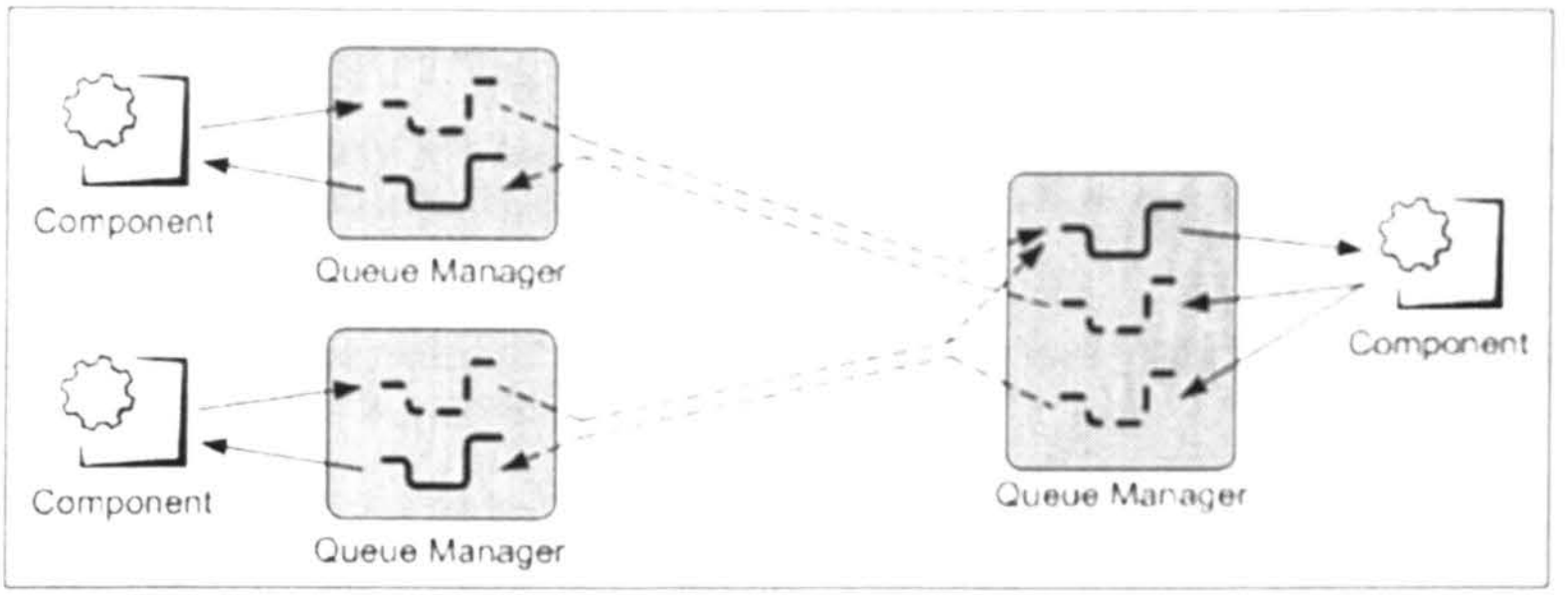
Pre zapojenie dvoch vzájomne komunikujúcich komponentov, ktorí sú však od seba vzdialení je možné použiť dva Queue Managery, cez ktoré vzájomná komunikácia preteká. Je však nutné vytvoriť špecifické fronty odkazujúce sa na vzdialené fronty pre vzdialené prepojenie. Tieto fronty sa nakonfigurujú a pomocou transmission queues a kanálov sa z týchto „virtuálnych" front správy automaticky prenesú do vzdialených cieľových front, kde sú spracované cieľovým komponentom. Toto riešenie slúži aj ako určitá záloha pri výpadku vzdialeného Queue Managera, kedy sa posielané správy nestratia a sú uchované v lokálnom Queue Managerovi v jeho transmission queues.

Toto zapojenie je taktiež používané pri komunikácii s vysoko zabezpečenými servermi, uloženými napríklad v demilitarizovanej zóne. V takom prípade je možné využiť fakt, že prepojenie Queue Managerov môže inicializovať ktorýkoľvek z nich.



Obrázok 23 : Komunikácia vzdialených bodov

Pri napojení viacerých „konzumentov" služby je potrebné definovať jednotlivé fronty a odpovedajúce odkazy na tieto fronty pre úspešnú komunikáciu.

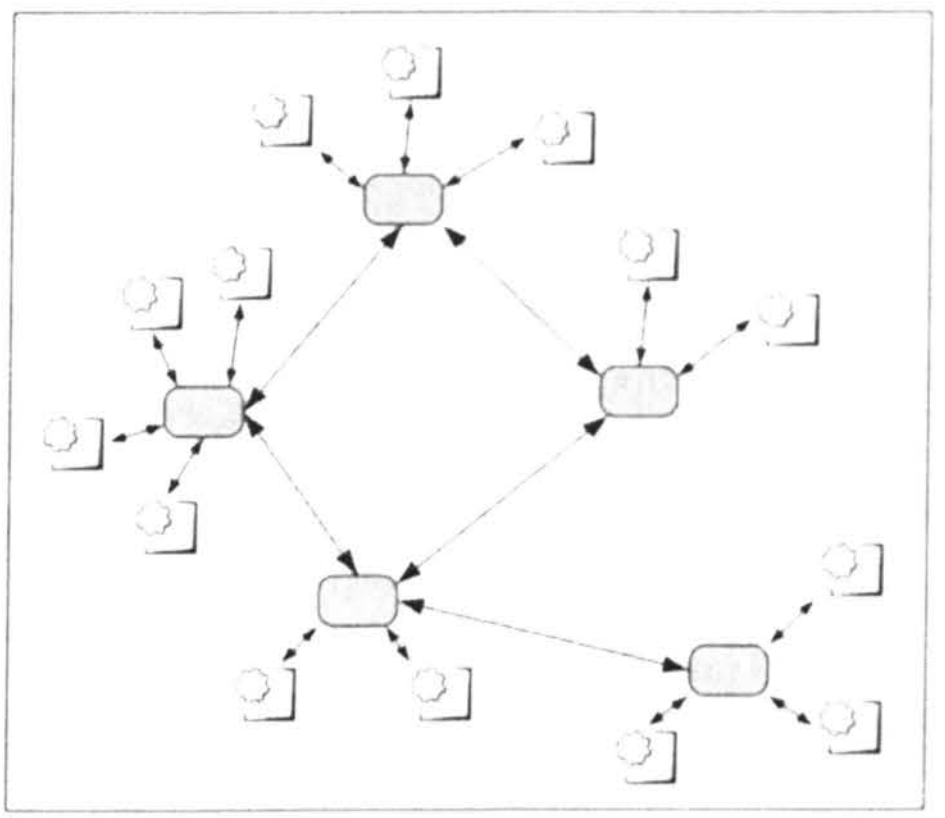


Obrázok 24 : Prepojenie viacerých konzumentov

Počet zapojených komponentov do celkovej topológie závisí len na požiadavkách riešenia. Platia jednoduché pravidlá:

- Každý komponent prijímajúci správy musí mať pre tento účel frontu, do ktorej tieto správy prijíma.
- Cez jednu frontu putujú správy len jedným smerom.
- Fronta určuje len svojho príjemcu, odosielateľa nie. Do jednej fronty môžu vkladať svoje požiadavky viacerí „konzumenti". V takom prípade je v tele každej správy identifikácia fronty, do ktorej má byť vložená odpoveď.

Postupným budovaním je možné vytvoriť dostatočne robustnú infraštruktúru schopnú splniť požiadavky na vzájomnú komunikáciu vnútorných bodov konfederácie.



Obrázok 25 : Príklad riešenia založenom na TIF-e

3.4.2. Obsluha správ

Vrstva spravujúca správy je v rámci architektúry TIF-u postavená nad transportnou vrstvou a slúži na uľahčenie a zjednotenie práce s aplikačnými správami prenášanými transportnou vrstvou.

Táto vrstva umožňuje serializovať a deserializovať aplikačné správy „do" a „z" podoby v akej sú prenášané transportnou vrstvou.

Prenášané správy majú dva možné podporované formáty

- Formát XML dokumentu
- Formát pevných dĺžok

Oba formáty správ sú určené na prenos tých istých dát. Líšia sa len formátom prenášanej správy.

Správa vo formáte XML dokumentu je textový reťazec obsahujúci validný XML dokument definovanej štruktúry. Jednotlivé prenášané dáta sú v tomto XML dokumente reprezentované ako hodnoty tagov tohto dokumentu. Jednotlivé tagy popisujú parametre správy, ktoré reprezentujú.

```
<?xml version="1.0" encoding="..."?>
<message>
    <head>
        [redacted]
    </head>
    <body>
        [redacted]
    </body>
</message>
```

Obrázok 26 : Správa vo formáte XML

Správa vo formáte pevných dĺžok je textový reťazec obsahujúci dáta lineárne zoradené za sebou bez akýchkoľvek oddeľovačov. Vytváranie a čítanie týchto správ je možné len za pomoci znalosti presnej definície správy, to znamená jednotlivých dátových typov, dĺžok a poradia prenášaných dát oboma komunikujúcimi stranami.

Použitie tohto formátu je určené pre systémy s extrémnymi požiadavkami na výkonnosť. Vznikol potrebou zjednodušenia parsovania správ v natívnych implementáciách komponentov na tomto systéme a nutnosti maximálnej rýchlosti spracovávania správ. Väčšinou sa však pri vzájomnej komunikácii používajú správy formátu XML.

Dáta prenášané správou sa rozdeľujú na dve oddelené logické časti

- Hlavička správy
- Telo správy

Hlavička správy obsahuje dáta povinné pri každej vzájomnej komunikácii. Patria sem položky, ktoré identifikujú samotnú správu, komunikujúce strany, časové razítka a rôzne systémové položky. Dáta hlavičky popisujú samotnú správu.

Telo správy obsahuje prenášané aplikačné dáta. Tieto aplikačné dáta sú jednoznačne popísané definíciou služby, ktorú daná správa obsluhuje. Táto definícia popisuje jednotlivé parametre požiadavky i odpovede, ich dátové typy ako aj jednotlivé veľkosti a presnosti dát.

3.4.3. QoS

Vrstva ktorá pridáva určitú množinu rozširujúcich funkčností nad spodnými vrstvami. Táto vrstva poskytuje nasledujúcu pridanú hodnotu :

- Identifikácia komunikujúcich strán
- Overovanie identity komunikujúcich strán
- Sémantická kontrola správy
- Error handling
- Logging
- Alerting
- Load Balancing & Error handling

Identifikácia komunikujúcich strán je založená na automatickom doplňovaní atribútov identifikujúcich odosielateľa správy do odchádzajúcich správ. Tieto identifikačné údaje následne slúžia na identifikovanie jednotlivých užívateľov služieb backendu ako aj pri logovaní požiadavkou.

Digitálne podpisovanie správ slúži pre overenie identity vzájomne komunikujúcich strán. Overenie identity protiľahlej komunikujúcej strany je dôležité pri výmene citlivých údajov pre zabránenie neoprávneného využívania služieb neznámym konzumentom.

Vo frameworku sa využíva symetrická šifra jednoznačne definovaná pre každú komunikujúcu dvojicu strán. Touto šifrou sa digitálne podpisujú správy prenášané frameworkom.

Sémantická kontrola správ slúži na overenie obsahovej správnosti vymieňaných správ. Kontrola správnosti je založená na kontrole jednotlivých parametrov správy. Kontroluje sa dátový typ, veľkosť a poprípade presnosť jednotlivých parametrov.

Error handling poskytuje mechanizmus jednotného a presne definovaného ošetrenia chybových stavov vzniknutých na jednotlivých komponentoch frameworku alebo pripájaných frontendoch a backendoch. Tento mechanizmus zahrňuje presnú identifikáciu chyby, jej kategorizáciu a následné spracovanie pomocou ďalších QoS ako sú Logging a Alerting. Error handling zohľadňuje pasívnosť / aktívnosť správy.

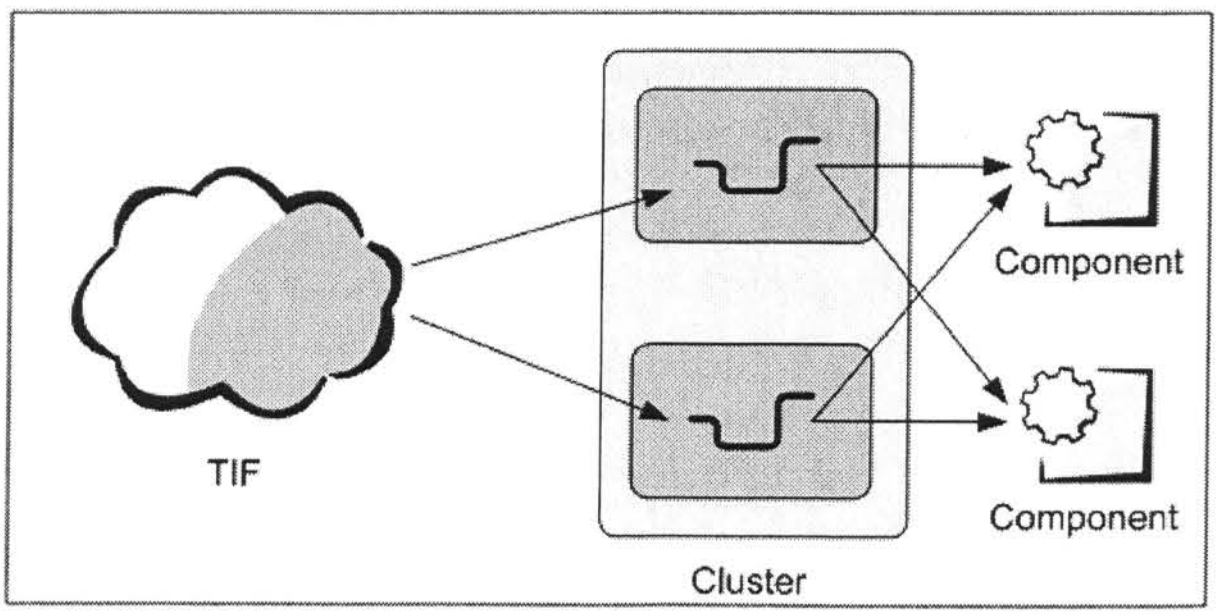
Logging je nástroj na sledovanie činnosti jednotlivých komponentov frameworku. Sledovať je možné používanie jednotlivých backendov, transportnej vrstvy ako aj volania jednotlivých frontendov. Sledujú sa relevantné udalosti vzniknuté počas behu systému. Vďaka logovaniu je možné spätne dohľadať dianie na jednotlivých komponentoch a poprípade i dohľadať okolnosti vzniku prípadnej chyby.

Alerting je mechanizmus hlásenia vzniknutých chybových stavov na príslušné definované miesta, ktoré zaručia návrat systému do konzistentného stavu a postarajú sa o zistenie príčiny chyby a obmedzenie jej ďalšieho vzniknutia. Adresátom alertingu je väčšinou dohľadové centrum a použitými nástrojmi sú špeciálne chybové fronty transportnej vrstvy alebo emailová notifikácia o vzniknutej chybe.

Error handling, logging a alerting sú vysoko účinné metódy pre správu, zotavovanie sa a následnú opravu vzniknutých chýb budovaného riešenia.

Load Balancing a Error Handling sú v TIF-e primárne riešené s využitím možností, ktoré poskytuje produkt WebShere MQ. Využívajú sa tieto mechanizmy :

- Možnosť nasadenia viac aplikácií (alebo viac výpočtových vlákien jednej aplikácie), ktoré vyberajú správy z jednej fronty a následne ich spracovávajú. Tieto aplikácie môžu bežať na rôznych serveroch. Za normálnych podmienok dochádza k rozprestretiu záťaže a v prípade výpadku jednej z aplikácií druhá funguje ako záloha zaisťujúca spracovanie všetkých požiadaviek.
- Možnosť distribúcie požiadaviek na väčší počet MQ Queue Managerov s využitím MQ Clusteringu. Pokiaľ je určitá fronta publikovaná do clusteru viacerých Queue Managerov, MQ infraštruktúra automaticky rozprestrie požiadavky do všetkých dostupných publikovaných front. Vďaka tomu dochádza k rozprestretiu záťaže na väčší počet Queue Managerov, ktorí môžu bežať na rôznych miestach a poskytovať týmto spôsobom zálohu riešenia.
- Kombinácia predchádzajúcich dvoch prístupov sa nakoniec osvedčila. Požiadavky sú smerované do dvoch Queue Managerov na rôznych serveroch, nad každou frontou následne pracujú dve aplikácie, každá na zvláštnom serveri. Vďaka tomu dochádza k minimalizácii rizika výpadku ľubovoľného Queue Managera, ľubovoľného serveru i aplikácie.



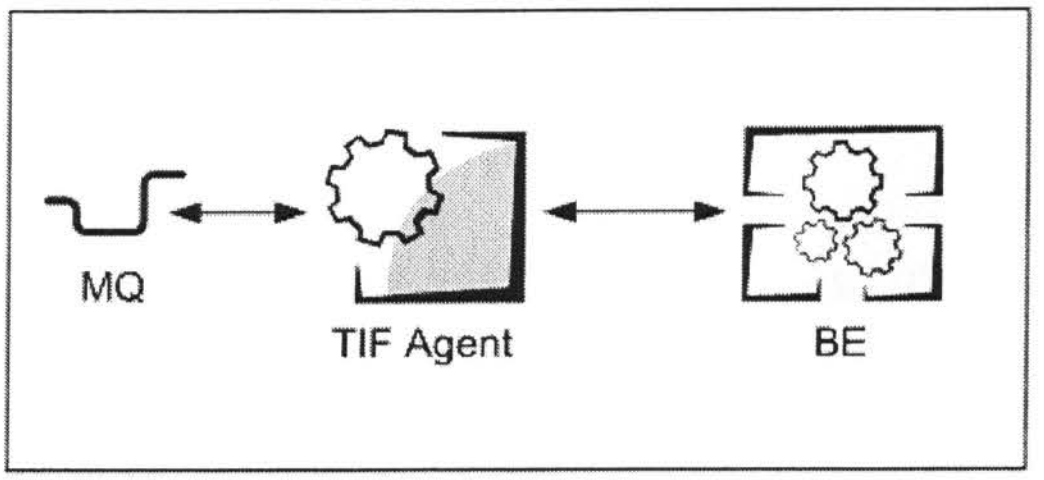
Obrázok 27 : TIF Load Balancing & Backup

3.4.4. Agenti

Agenti sú univerzálne komponenty tvoriace vrchol vrstvy architektúry, ktoré slúžia na aktívne pripojenie existujúcich backendových komponentov do vytváraného systému.

Tvoria posledný blok pred samotným integrovaným backendom. Vo väčšine prípadov sú pripojení jedna k jednej k danému backendu. Agenti tvoria bránu k danému backendu.

Agenti slúžia na jednotné integrovanie služieb backendu do servisne orientovanej architektúry. Služba backendu je následne prístupná jednotným spôsobom nehľadiac na skutočný komunikačný protokol backendu. Táto brána transformuje volanie unifikovanej služby na volanie služby backendu.



Obrázok 28 : Zapojenie backendu pomocou agenta

S daným backendom komunikujú za pomoci natívneho rozhrania daného backendu. Jednotlivé natívne rozhrania s implementované rôznymi Pluginmi, ktoré sa po nakonfigurovaní „zasunú" do Agenta, čím je umožnená komunikácia Agenta s backendom pomocou natívneho rozhrania.

Tento spôsob stavby Agentov umožňuje jednoduché rozšírenie možností frameworku o podporu ďalších typových i atypických druhov backendov podľa potrieb zákazníka.

Pluginy riešia všetky otázky ohľadom prepojenia backendu

- Zabezpečenie natívnej komunikácie s backendom
- Transformovanie natívnych dátových typov do dátových typov frameworku
- Obsluhu chybových stavov backendu
- Transakčnosť backendov

Prehľad existujúcich Pluginov umožňujúcich integrovanie natívnych backendov

Názov Pluginu	**Popis Pluginu**
TIF Plugin for Oracle	Plugin pre natívny prístup do serverovej aplikácie postavenej nad DB Oracle.
TIF Plugin for SQL Server	Plugin pre natívny prístup do serverovej aplikácie postavenej nad DB MS SQL Server.
TIF Plugin for ODBC	Plugin pre natívny prístup do serverovej aplikácie postavenej nad databázou dostupnou cez ODBC.
TIF Plugin for COM	Plugin pre natívny prístup do serverovej aplikácie postavenej bázy technológie Microsoft COM.
TIF Plugin for EJB	Plugin umožňujúci volať štandardné serverové komponenty v rámci J2EE aplikačného serveru.
TIF Plugin for WebServices	Plugin pre natívny prístup do serverovej aplikácie s rozhraním na bázy Web Services.

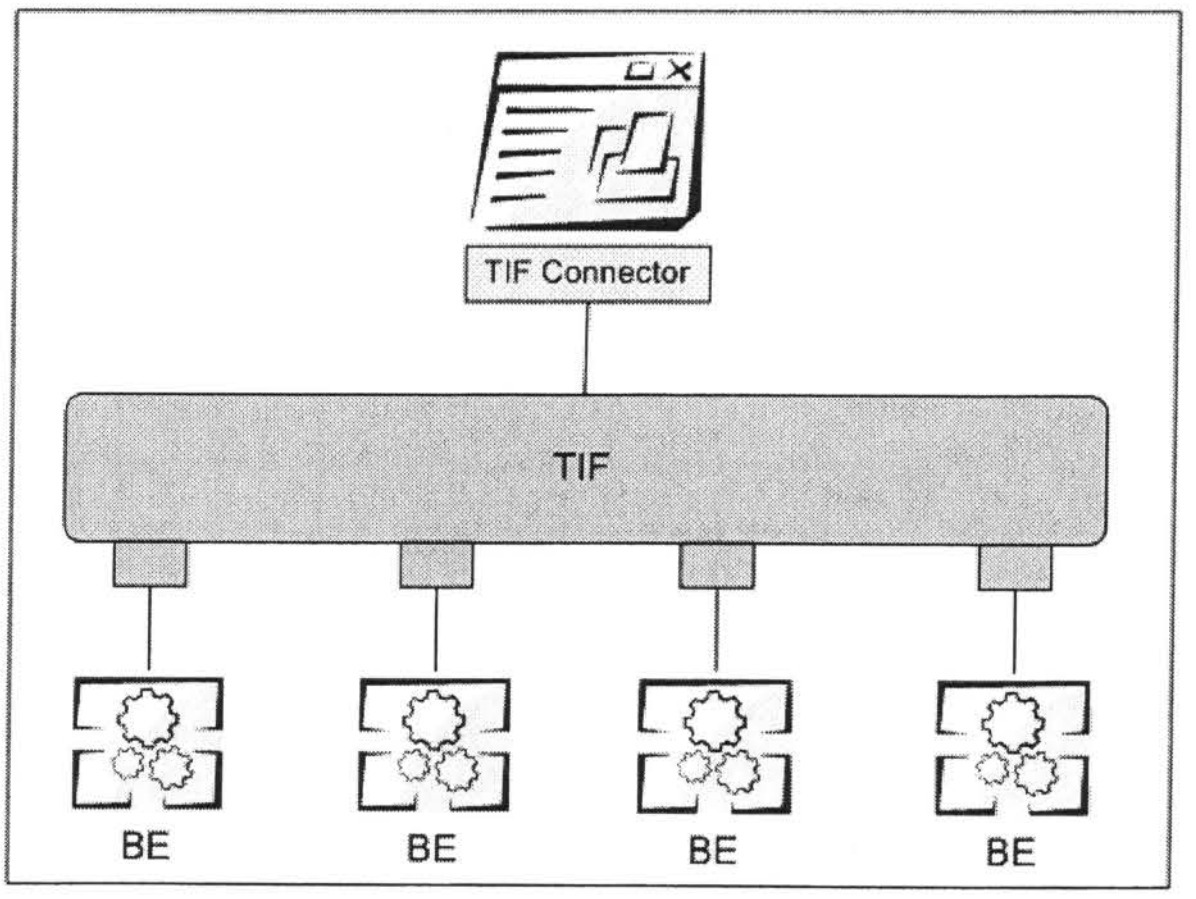
TIF Plugin for S2S	Špecifický plugin pre aplikáciu S2S.
TIF Plugin for iFlex	Špecifický plugin pre aplikáciu iFlex Flexcube@Connect.
Sanchez Plugin	Špecifický plugin pre bankovú aplikáciu Sanchez Profile.
Plugin for Optical Archive	Špecifický plugin pre DMS aplikáciu eiStream Enterprise.
TIF Plugin for IBM Content Manager	Špecifický plugin pre DMS aplikáciu IBM Content Manager.

3.4.5. Konektory

Klientské rozhrania umožňujúce zapojenie frontendových komponentov do komunikačnej infraštruktúry.

Sú distribuované ako knižnice jednoducho použiteľné z kódu klientskej aplikácie. Volanie jednotlivých služieb je veľmi jednoduché a odtieňuje klientskú aplikáciu od nutnosti znalosti princípov fungovania frameworku. Poskytuje taktiež všetky pridané hodnoty frameworku.

Volanie služieb je pomocou konektorov zjednodušené na maximálnu možnú úroveň.



Obrázok 29 : Zapojenie TIF Connector-u

Umožňuje vygenerovanie „Proxy" tried, ktoré vďaka predgenerovaným settovacím a gettovacím metódam urýchľujú vývoj klientskej aplikácie a zjednodušujú používanie Konektoru pri zmene služieb.

Momentálne sú podporované tri jazykové mutácie Konektorov

- TIF Connector for Java
- TIF Connector for C
- TIF Connector for COM

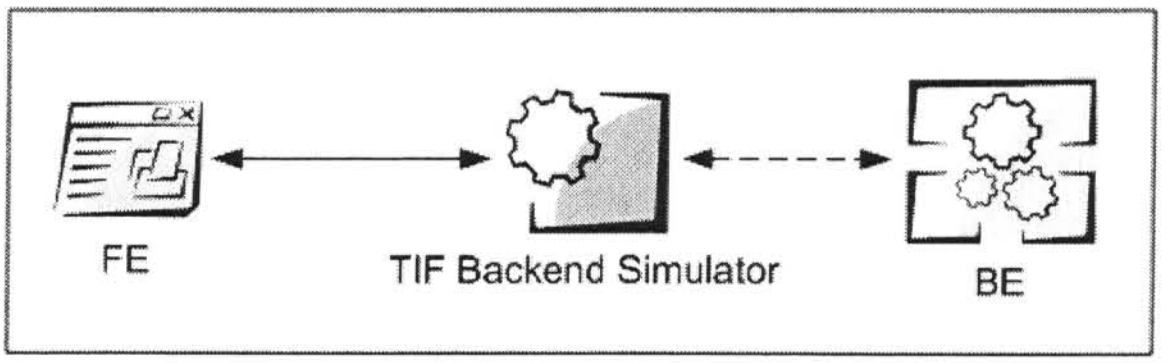
Tieto tri jazykové mutácie zaručujú dostatočne širokú využiteľnosť Konektoru v rôznych programovacích jazykoch a tým pádom poskytujú dostatočné možnosti integrácie frontendových aplikácií do riešenia.

3.4.6. Aktívne komponenty

Aktívne komponenty je množina komponentov integračného frameworku umožňujúcich riešiť rôzne integračné úlohy. Momentálne je k dispozícii množina komponentov

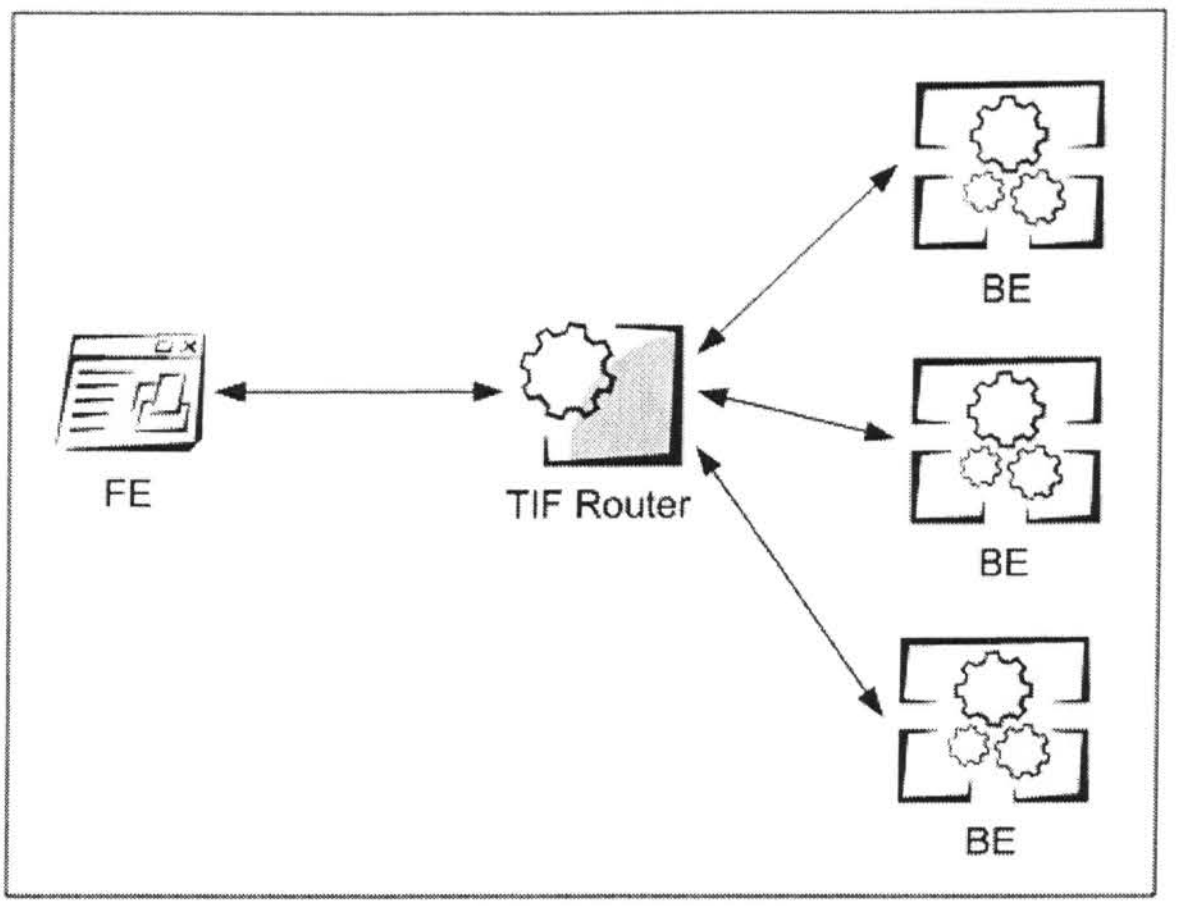
- TIF Backend Simulator
- TIF Router
- TIF Inspector
- TIF Publisher
- TIF B2BGateway

TIF Backend Simulator – komponent na simulovanie odpovedí správ podľa obsahu (Content Based Simulation). TIF Backend Simulator na základe konfigurácie vyhodnocuje obsah správ a buď na danú správu simuluje odpoveď, alebo ju prepošle na cieľový backend, ktorý už danú správu obslúži. Tento komponent je vhodný pri vývoji nového backendu za situácie, keď je potreba volať služby ešte neimplementované daným backendom.



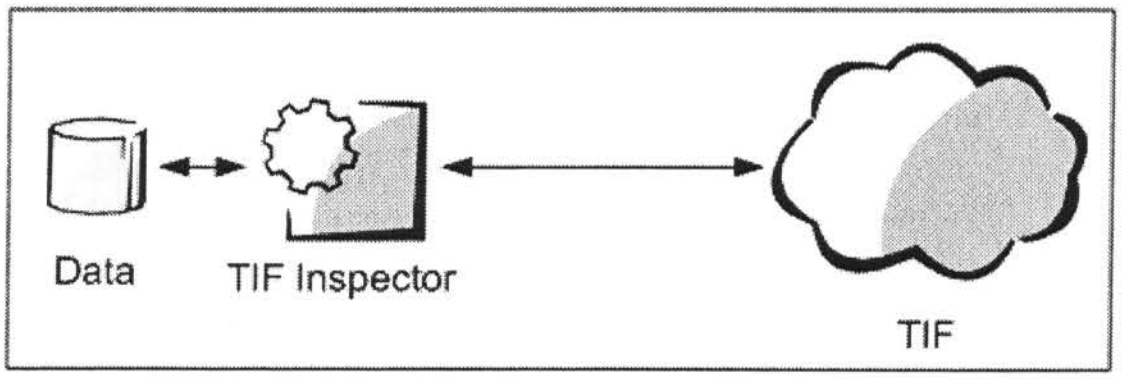
Obrázok 30 : Zapojenie TIF BESIM-u

TIF Router – komponent vykonávajúci Content Based Routing. Prijímané správy, na základe definovaných pravidiel popisujúcich obsah správy, preposiela ďalej na príslušné cieľové backendy.



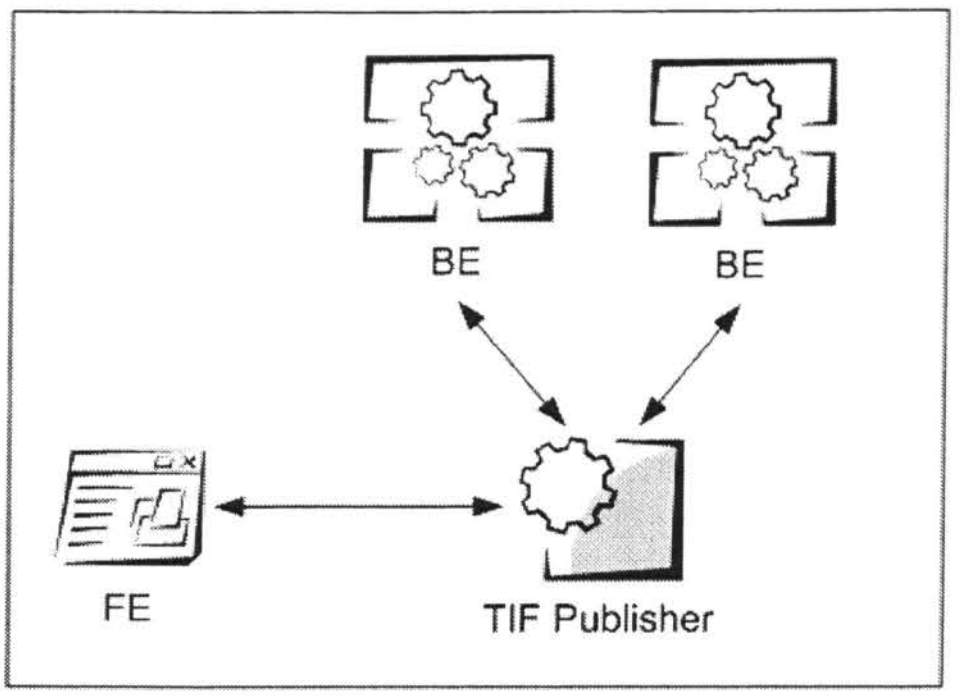
Obrázok 31 : Zapojenie TIF Routera

TIF Inspector – komponent schopný aktívne načítávať dáta z daných backendov a následne takto získané dáta vo formáte volaní špeciálnych služieb distribuovať do integračnej infraštruktúry, v rámci ktorej môžu byť dáta spracované. TIF Inspector umožňuje riešiť replikačné úlohy.



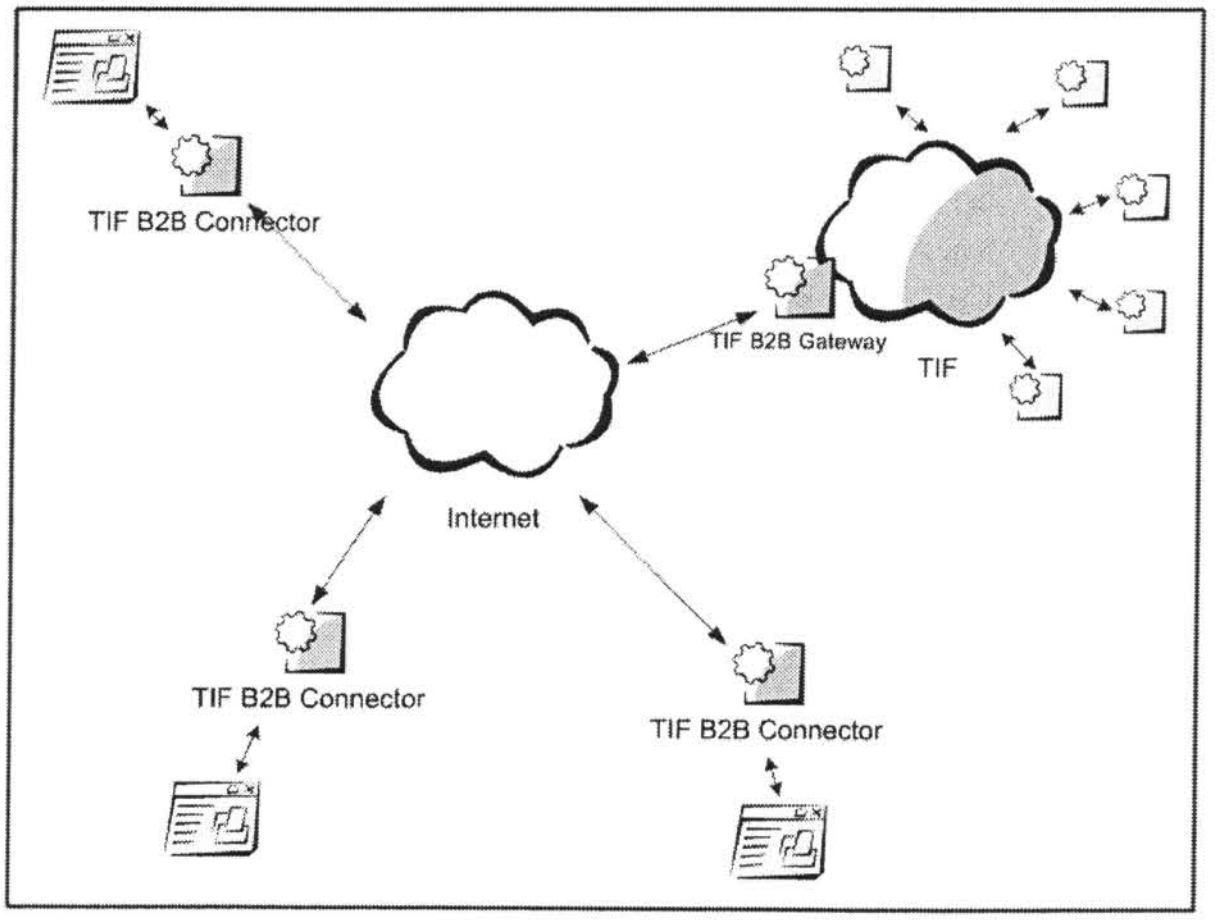
Obrázok 32 : Zapojenie TIF Inspectoru

TIF Publisher – Aktívny komponent vykonávajúci mikroworkflow nad prichádzajúcimi správami. Jednotlivé mikroworkflow procesy sú definované za pomoci zásuvníkových Java tried implementujúcich špecifickú funkčnosť. V rámci tohto mikroworkflow je možné takisto volať ďalšie komponenty a využívať služby nimi ponúkané.



Obrázok 33 : Zapojenie TIF Publisheru

TIF B2BGateway – Komponent integračného frameworku slúžiaci na prepojenie konfederácie s jej obchodnými partnermi. Pomocou TIF B2BGateway je možné poskytnúť definované služby obchodným partnerom. Tieto služby majú presné definície a tvoria rozhranie poskytované organizáciou externým partnerom. TIF B2B Gateway kladie vysoké požiadavky na bezpečnosť ponúkaného rozhrania a zároveň jednoduchosť použitia. Preto je súčasťou TIF B2B Gateway aj komponent TIF B2B Connector, ktorý slúži na jednoduchú a bezpečnú komunikáciu s TIF B2B Gateway zo strany partnerskej organizácie.



Obrázok 34 : Použitie TIF B2B Gateway a TIF B2B Connectoru

3.4.7. Nástroje

Nástroje ponúkané integračným frameworkom tvoria množinou podporných programov pre uľahčenie a zjednodušenie práce s infraštruktúrou a integračným frameworkom samotným. Množinu nástrojov ponúkaných TIF-om je možné rozdeliť na nástroje pre vývoj a nástroje pre podporu samotného behu riešenia.

- Vývoj
  - TIF Tester
  - TIF User Rules Editor
- Beh
  - TIF Monitor
  - TIF Remote Controll
  - TIF Security Utils

TIF Tester – program pre testovanie funkčností ponúkaných jednotlivými backendami. Nástroj poskytuje grafické rozhranie pre zjednodušenie ovládania a efektívnu prácu. Časté využitie tohto nástroja spočíva pri vývoji nových služieb na backendoch.

TIF User Rules Editor – nástroj pre spravovanie pravidiel pre Content Based Routing a Content Based Simulation. Grafické rozhranie zjednodušuje prácu s týmito pravidlami, umožňuje definíciu užívateľov oprávnených spravovať pravidlá.

TIF Monitor – aplikácia s webovým grafickým rozhraním určená pre monitoring transportnej vrstvy. Pomocou tohto nástroja je možné monitorovať vyťaženie jednotlivých prenosových liniek transportnej vrstvy, zisťovať preťaženosť jednotlivých front. Zo zistených údajov je možné poprípade doladiť konfiguráciu transportnej vrstvy pre optimalizovanie zo strany požiadaviek na beh systému.

TIF Remote Controll – nástroj pre vzdialené sledovanie činnosti bežiacich Agentov. Nástroj umožňuje sledovať stav komponentov, štatistiky vykonávania jednotlivých služieb, chybovosť behu komponentov a umožňuje aj daný komponent zastaviť alebo reštartovať.

TIF Security Utils – nástroj pre zjednodušenie administrácie zabezpečenia jednotlivých komponentov infraštruktúry.

3.5. Definície služieb

Pre vytvorenie integračného riešenia je nutné určitým jednotným spôsobom definovať rozhrania jednotlivých služieb ponúkaných systémom. Táto definícia musí obsahovať dostatočné množstvo metadát pre zaručenie použiteľnosti definície v rámci integračného frameworku.

Konektory na základe definície služieb umožňujú volanie danej množiny služieb a sú schopné vytvoriť správnu správu volajúcu danú službu. Na základe tej istej definície konektor spracováva aj prípadnú odpoveď na požiadavku.

Táto definícia je dôležitá aj pre koncových Agentov, aby boli schopní zavolať správne služby na backendoch a korektne pracovať so správnymi dátovými typmi parametrov volania a vytvoriť odpovedajúcu odpoveď, ak je požadovaná.

Aktívne prvky TIF-u vďaka definícia služieb dokážu korektne pracovať s jednotlivými správami a vykonávať na nich definovanú funkčnosť.

Definície služieb preto obsahujú nasledujúce položky

- Popis služby – názov, verzia, popisok, príznak aktívnosti

- Popis jednotlivých parametrov služby – smer, dátový typ, presnosť, veľkosť, povinnosť
- Informácie pre Agenta umožňujúce správne spracovanie služby na backende – napr. natívny dátový typ backendu, mapovanie na parametre služby atď.

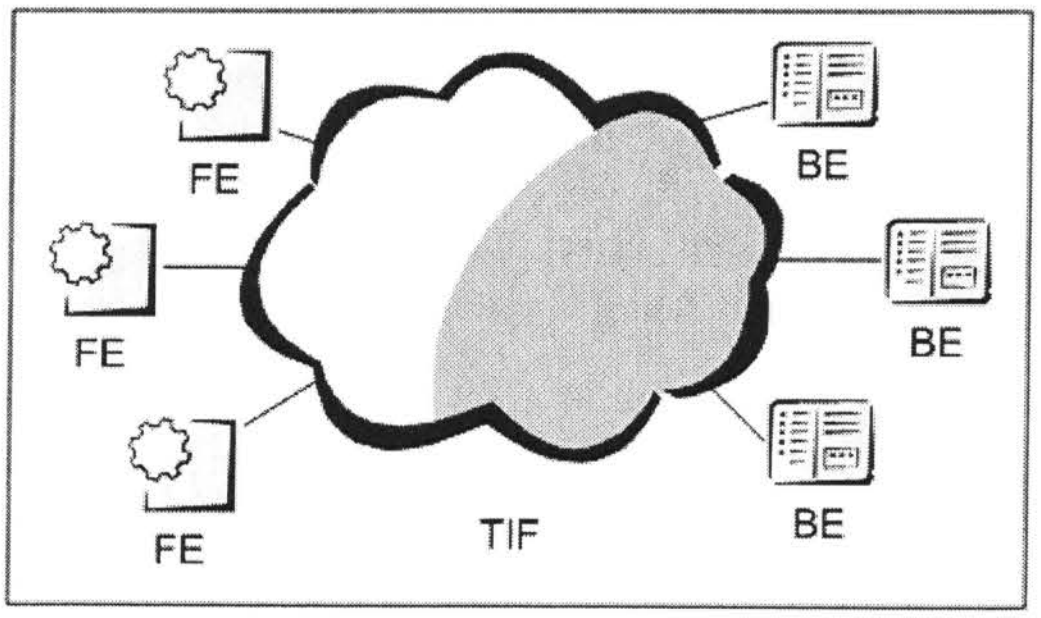
Definície služieb sú uložené do dvoch možných dátových úložisko

- XML súbor – súbory presnej štruktúry definujúce všetky potrebné atribúty služby pre jej úspešné použitie. Táto konfigurácia je uložená väčšinou u komponentu využívajúceho danú definíciu. Umožňuje definovať pre daný komponent presný zoznam služieb.
- DB – definície jednotlivých služieb sú uložené v centrálnej databáze. Táto databáza eviduje u každej definície identifikátor backendu, ktorý danú službu poskytuje. Poskytuje možnosť centrálnej správy definícií na jednom mieste. Komponenty však k týmto definíciám nepristupujú, databáza sa využíva na generovanie XML definičných súborov. Databáza tvorí centrálne úložisko definícií služieb (metadata repository), ale nie je priamou súčasťou infraštruktúry a nevytvára riziko pri výpadkoch.

3.6. Použitie

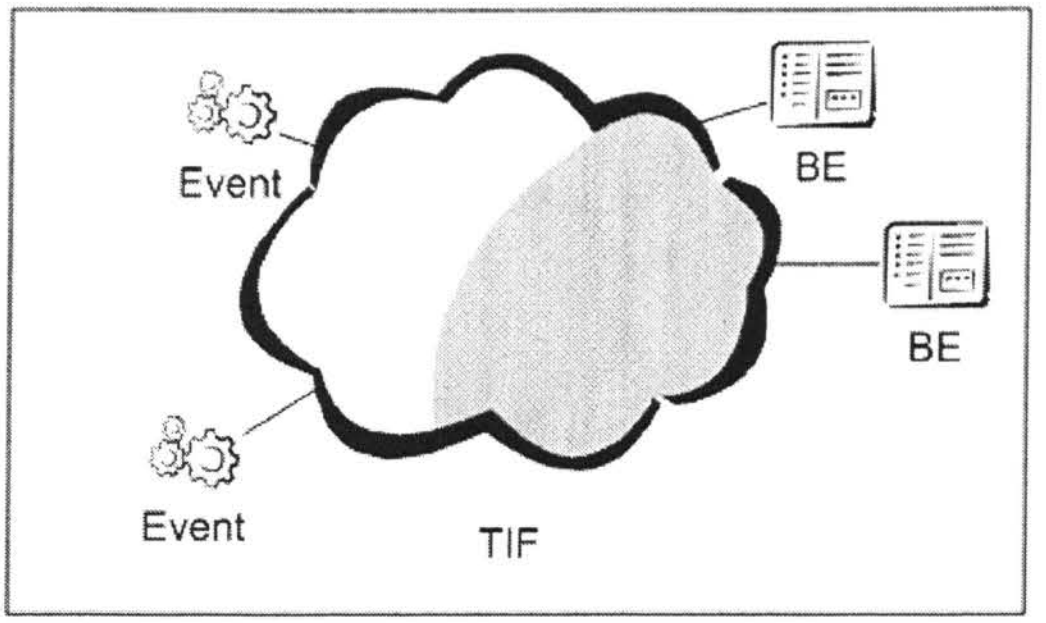
TIF slúži primárne na flexibilné prepojenie servisne orientovaných komponentov konfederácie (SOA). Tieto komponenty môžu byť heterogénneho charakteru. Poskytuje možnosť rýchleho a jednoduchého rozšírenia systému o novo integrované komponenty. Skrátene povedané „Configuration-based integration of heterogeneous shared services".

Umožňuje taktiež vytváranie kompozitných služieb používajúcich pre svoj beh volanie ďalších dostupných služieb riešenia.



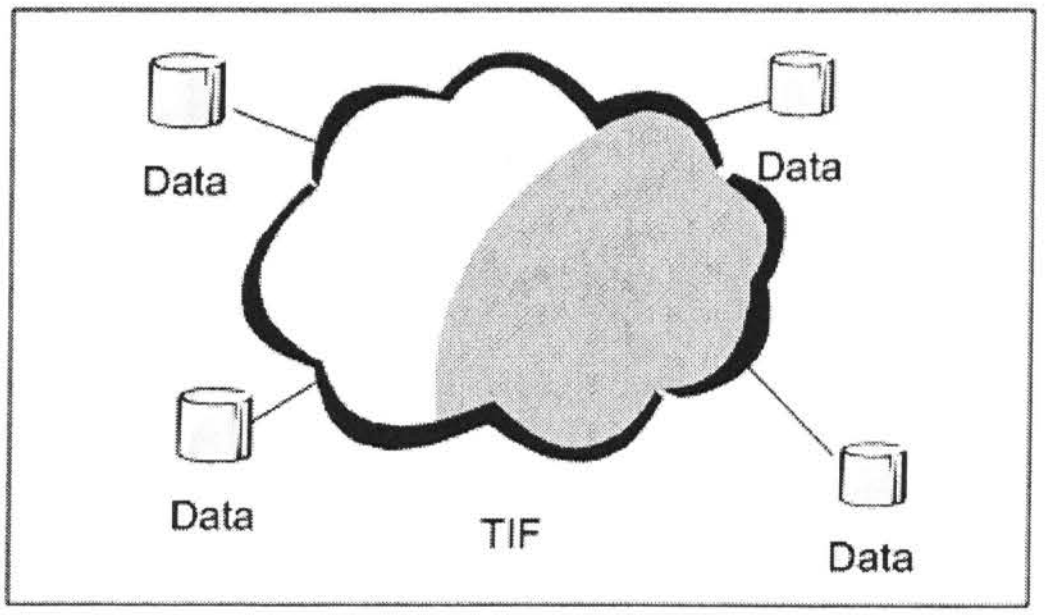
Obrázok 35 : Použitie TIF pre riešenie SOA

Ďalším integračným scenárom riešiteľným pomocou TIF-u je replikácia dát založená na udalostiach (Event Driven Architecture - EDA). Integračný scenár rieši problém nutnosti rýchlej distribúcie udalosti v systéme.



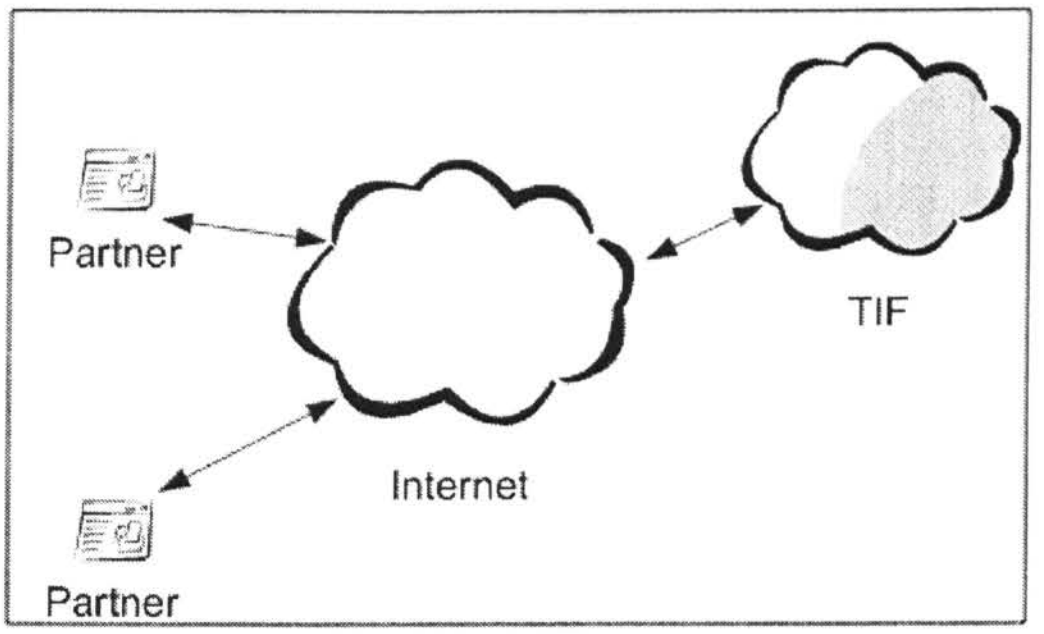
Obrázok 36 : Použitie TIF pre riešenie EDA

TIF taktiež umožňuje riešiť problémy Extract, Transform and Load (ETL). Tento pojem zahŕňa prenos objemných dát väčšinou vo forme dávok, ktoré sa zreplikujú z jedného dátového zdroja do iného (viacerých) dátového zdroja. Pri prenose dát môže dochádzať k ich transformácii.



Obrázok 37 : Použitie TIF pre riešenie ETL

TIF podporuje integráciu Business to Business (B2B). Tento integračný scenár rieši problém prepojenia spoločnosti s jej obchodnými partnermi a sprístupnenie určitej množiny služieb pre obchodných partnerov.



Obrázok 38 : Použitie TIF pre riešenie B2B

TIF poskytuje možnosť riešenia všetkých integračných scenárov v rámci jednej inštancie TIF-u. Riešenie týchto scenárov pracuje väčšinou so zdieľanou množinou zdrojov (infraštruktúry TIF-u i infraštruktúry organizácie).

3.7. Predpoklady pre možné použitie

Pre úspešnú integráciu komponentov konfederácie za pomoci integračného frameworku TIF je nutné splniť určité predpoklady.

3.7.1. Predpoklady kladené na backendy

- Backend musí poskytovať rozhranie v natívnom jazyku, cez ktoré je umožnené volať poskytované služby.
- U integrovaných komponentov sa predpokladá existencia servisných služieb pre možnosť monitorovania stavu a identity integrovaného komponentu.
- U poskytovaných služieb sa predpokladá existencia servisných parametrov, ktoré vypovedajú o výsledku volania služby a poskytujú podrobné informácie o stave vzniknutých udalostií pri spracovaní požiadavky v komponente.
- Tieto požiadavky vznikli za účelom zvýšenia úrovne QoS integračného frameworku.

3.7.2. Predpoklady kladené na frontendy

- Schopnosť integrovať a volať TIF Connector slúžiaci na volanie služieb poskytovaných systémom

3.7.3. Predpoklady kladené na dátové zdroje

- Poskytnutie rozdielových dát, ktoré sú aktívne načítané a posielané.
- Možnosť zapisovania (ukladania) výsledku replikovania dát do dátového zdroja.
- Schopnosť generovania udalostí pre úspešné riešenie EDA.

3.8. Vlastnosti TIF

3.8.1. Základné vlastnosti

- TIF je Enterprise Service Bus pre prepojenie, sprostredkovanie a spravovanie interakcie medzi koncovými bodmi konfederácie.
- TIF je middleware. Je to software pre vzájomné prepojenie softwarových komponentov za účelom podpory distribuovaných aplikácií.
- Integračný framework založený na výmene správ pomocou asynchrónnych front.
- Formát správ je interný, definovaný frameworkom. Správy majú charakter XML dokumentov.
- Systém integrovaný pomocou TIF-u spĺňuje charakteristiky Service Oriented Architecture.
- TIF poskytuje volania jednotlivých backendov ako služby ponúkané systémom.
- Definície jednotlivých služieb majú formát XML súborov popisujúcich službu a jej jednotlivé parametre.

3.8.2. Ďalšie vlastnosti

- Podpora synchrónnych správ – celkovo sa systém aj napriek asynchrónnemu posielaniu správ môže javiť ako synchrónny. Táto vlastnosť je dosiahnutá vďaka koncovým bodom systému, ktoré zaobaľujú asynchrónne posielanie správ a pomocou vhodného párovania správ sa komunikácia navonok javí ako synchrónna.
- Využíva konektory pre pripojovanie existujúcich aplikácií do systému – pre typické aplikácie poskytuje middleware sadu konektorov, ktoré dokážu komunikovať s týmito aplikáciami pomocou ich natívneho rozhrania. Tieto konektory prekladajú dátové formáty koncových aplikácií do interného formátu a opačne.
- Štandardizovaná bezpečnosť – middleware pre internú komunikáciu medzi uzlami siete používa štandardizovanú bezpečnosť na úrovni transportnej i aplikačnej vrstvy. Pre komunikáciu s konkrétnymi koncovými aplikáciami je bezpečnosť zaručená konkrétnymi metódami zabezpečenia jednotlivých typov koncových aplikácií.
- Validácia dát – pre tok dát v systéme je dôležitá i určitá kontrola dát a to nielen kontrola formátu dát, ktorá je dôležitá hlavne pre transformáciu dátových formátov, ale aj kontrola správnosti rôznych dátových typov. Kontrola dátových typov môže mať za dôsledok odľahčenie transportnej vrstvy, keďže chybné požiadavky sú odfiltrované už pri vznesení požiadavky na middleware a nie sú propagované až do koncových uzlov, kde by pri ich validácii zistené chyby museli byť propagované späť. Validácia dátových typov prebieha na základe definícií služieb kde sú uvedené práve aj dátové typy posielaných dát.
- Správa chýb – pre správny chod systému je dôležitá správa chýb. Spravovanie chýb je implementované na rôznych úrovniach. Od propagácie chyby koncového bodu späť do volajúcej aplikácie až po evidovanie chýb jednotlivých koncových systémov a následné preposielanie týchto chýb administrátorom daných systémov.
- Podpora Event handling – middleware podporuje publikovanie udalostí vznikajúcich na koncových aplikáciách. Konektory aktívnym poolovaním vyhľadávajú propagované udalosti, ktoré sú pomocou aktívneho prvku middlewaru Publisher implementujúceho business logiku publikované do koncových aplikácií systému za pomoci volania služieb týchto aplikácií.

- Inteligentný routing založený na obsahu správ –TIF podporuje content based routing za pomoci aktívneho komponentu, cez ktorý preposielané dáta prechádzajú.
- Centrálna správa definícií rozhraní – umožňuje centrálne spravovať definície rozhraní ponúkaných služieb. Zároveň poskytuje alternatívu pre definovanie rozhraní jednotlivých služieb ako súčasť konfigurácie jednotlivých komponentov TIF-u.

3.8.3. Charakteristiky ktoré TIF plne nespĺňa

- Podpora Publish / Subscribe – podpora tohto scenára posielania správ nie je primárne podporovaná. Pre vytvorenie aktívneho komponentu je ale možné jednoduchým modifikovaním komponentu Publisher vytvoriť komponent pre podporu Publish / Subscribe.

3.9. Porovnanie

Pre porovnanie základných charakteristík ESB produktov s TIF-om boli použité nasledovné produkty:

- BEA AquaLogic Service Bus 2.5 [58]
- Iona Artix 4.0 [59]
- Sonic ESB 7.0 [60]

Tieto produkty tvoria len ukážku možných ESB produktov ponúkaných rôznymi spoločnosťami ako napríklad BEA, IBM, Microsoft, Sap, iWay Software, Tibco a ďalšie.

Nasledujúca tabuľka poskytuje jednoduchý prehľad základných vlastností spomínaných produktov a ich porovnanie s TIF-om. Jednotlivé porovnávané aspekty tvoria základné vlastnosti ponúkané rôznymi implementáciami ESB.

Z uvedenej tabuľky vyplýva že prehľad základných vlastností rôznych implementácií ESB je celkom podobný a tieto implementácie sa líšia len v ponuke rozširujúcich QoS.

Zaujímavým faktom je však rozdielnosť jednotlivých implementácií ESB v štýle infraštruktúry. Na trhu sú produkty, ktoré je možné rodeliť do troch kategórií podľa štýlu infraštruktúry.

- WS – P2P infraštruktúra založená väčšinou na Webových službách tvoriacich koncové body infraštruktúry. (IONA, Cape Clear)
- MOM – P2P infraštruktúra založená na Message Oriented Middleware produktoch (Fiorano, Sonic)
- Centralizovaný server – Infraštruktúra založená na centralizovanom aplikačnom serveri. (IBM, BEA, Oracle, FusionWare, PolarLake)

TIF síce v ponuke QoS a možností transformácií mierne zaostáva za uvedenými produktmi, napriek tomu je možné ho označiť za ESB implementáciu, keďže poskytuje dostatočné možnosti vytvárania servisne orientovanej architektúry s pridanou hodnotou.

Produkt	Štýl infraštruktúry	Komunikácia	Transformácie	Validácia dát	Logging	Monitoring	Bezpečnosť	Ďalšie QoS
IONA Artix 4.0	WS	XML, SOAP / IIOP,JMS,HTTP, JMX	Automatická transformácia medzi podporovanými dát. formátmi	Schémy	Vysoká podpora	Status, štatistiky, BAM	Autentifikácia, Kryptovanie, Dig. podpisy, WS-Security, SSL, Certifikáty	Designer GUI, API, BPM, Error handling
Sonic ESB 7.0	MOM	WS / JMS,HTTP,FTP, SMTP, MQ	XPath,XQuery, XSLT	Schém	Štatistiky, chyby, výkonnosť, správy	Status, štatistiky	Autentifikácia, Kryptovanie, WS-Security, SSL, Certifikáty	Centralizovaná správa, auditing, reporting, alerting, triggers, BPEL
BEA AquaLogic Service Bus 2.5	Centralizovaný server	XML / JMS,HTTP,FTP, SMTP	XPath,XQuery, XSLT	Schémy, Služby	Štatistiky, chyby, výkonnosť, správy	Status, štatistiky, SLA	Autentifikácia, Autorizácia, Kryptovanie, Dig. podpisy, WS-Security, SSL, Certifikáty	Test Console, Centrálny Resource Cache, Error handling
TIF	MOM	XML / MQ	Automatická transformácia medzi podporovanými dát. formátmi, XSLT, XPath	Schémy	Štatistiky, chyby, výkonnosť, správy	Status, štatistiky	Autentifikácia, Kryptovanie, Dig. podpisy, SSL, Certifikáty	Error handling, reporting, alerting, testing, auditing

4. Záver

Architektúra orientovaná na služby je v tejto dobe široko využívaná a produkty pre jej podporu sú na trhu dostupné a ponúkajú dostatočnú kvalitu pre reálne nasadenie.

SOA pri dodržaní svojich princípov ponúka dostatočnú flexibilitu pre reagovanie na dynamické zmeny vyvíjajúcej sa spoločnosti. Náklady spojené so zavedením SOA nemajú síce okamžitú návratnosť, ale pri rozumnom nasadení a správnom navrhnutí architektúry je návratnosť citeľná.

Hlavným rizikom pri zavádzaní SOA je ale nutnosť porozumenia samotným princípom SOA. SOA vyžaduje špecifickú stratégiu manažmentu a marketingu. Výhody SOA sa dostavia iba pri správnom využití. [7]

Frameworky pre realizovanie servisne orientovanej architektúry v prostredí mimo e-komercie majú väčšinou charakter Enterprise Service Bus (ESB). Tento architektonický štýl okrem podpory SOA definuje i radu QoS, ktoré zvyšujú použiteľnosť i cenu samotných implementácií ESB. Z celej enklávy QoS sú však pre ESB mandatórne len niektoré.

V dnešnej dobe existuje mnoho implementácií SOA i ESB. Síce pochádzajú od rôznych výrobcov, majú však zrovnateľné vlastnosti. Práca nie je schopná pokryť celé portfólio produktov SOA, preto pre účely zhodnotenia výhod a nevýhod použitia SOA mimo e-komercie bol vybratý produkt Trask Integration Framework. Úspešnosť tohto produktu dokazuje, že aj v českých podmienkach je možné vytvoriť úspešnú implementáciu ESB, ktorá je schopná konkurencie produktom nadnárodných IT spoločností.

Literatúra

[1] http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#service

[2] http://www.humbertocervantes.net/servicesandcomponents.html

[3] http://www.trireme.com/whitepapers/design/components/component_design_how.html

[4] http://www.bitpipe.com/tlist/Composite-Applications-Software.html

[5] http://www.trireme.com/whitepapers/design/components/component_design_what.html

[6] Bruce Eckel : Myslíme v jazyku JAVA, Grada, 2001

[7] J. Král, M. Žemlička : Software Architecture for Evolving Environments, Workshop on Evolution of Software Systems in a Business Context, Budapešť 2005

[8] http://blogs.ittoolbox.com/eai/cto/archives/003526.asp

[9] http://webservices.sys-con.com/read/39953.htm

[10] http://webservices.sys-con.com/read/39908.htm

[11] http://en.wikipedia.org/wiki/Service-oriented_architecture

[12] http://www-128.ibm.com/developerworks/library/ws-soaintro.html

[13] http://www.looselycoupled.com/opinion/2005/erl-core-infr0815.html

[14] http://blogs.ittoolbox.com/eai/business/archives/soa-benefits-challenges-and-risk-mitigation-8075

[15] http://www-128.ibm.com/developerworks/webservices/library/ws-antipatterns/

[16] http://www.service-architecture.com/web-services/articles/web_services_definition.html

[17] Eric Newcomer : Understanding Web Services XML WSDL SOAP and UDDI, Addison Wesley Professional, 2002

[18] http://www-128.ibm.com/developerworks/webservices

[19] http://www.w3.org/TR/wsdl

[20] http://www.w3.org/TR/soap12-part1/

[21] http://www.oasis-open.org

[22] http://www.uddi.org/

[23] http://www-128.ibm.com/developerworks/webservices/library/ws-wisdom/

[24] http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-management1004.pdf

[25] http://www-128.ibm.com/developerworks/library/specification/ws-provis/

[26] http://www-128.ibm.com/developerworks/webservices/library/ws-secure/

[27] http://www-128.ibm.com/developerworks/library/specification/ws-secpol/

[28] http://www-128.ibm.com/developerworks/library/specification/ws-trust/

[29] http://www-128.ibm.com/developerworks/library/specification/ws-secon/

[30] http://www-128.ibm.com/developerworks/library/specification/ws-fed/

[31] http://www.webservicesarchitect.com/content/articles/apshankar04.asp

[32] http://www.serviceoriented.org/ws-privacy.html

[33] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

[34] http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[35] http://www-128.ibm.com/developerworks/webservices/library/ws-wsfl1/

[36] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[37] http://www.w3.org/TR/ws-cdl-10/

[38] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction

[39] http://www-128.ibm.com/developerworks/library/specification/ws-tx/

[40] http://www-128.ibm.com/developerworks/library/specification/ws-tx/#coor

[41] http://81.29.72.192/ws/content/view/full/52969

[42] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsroutspecindex.asp

[43] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsreferspecindex.asp

[44] http://www.beepcore.org/

[45] http://www-128.ibm.com/developerworks/webservices/library/ws-phtt/

[46] http://www-128.ibm.com/developerworks/library/specification/ws-eventing/

[47] http://www-128.ibm.com/developerworks/webservices/library/specification/ws-notification/

[48] http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-enumeration.pdf

[49] http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-transfer.pdf

[50] http://www-128.ibm.com/developerworks/library/specification/ws-add/

[51] http://www.w3.org/Submission/ws-messagedelivery/

[52] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsrmspecindex.asp

[53] http://developers.sun.com/sw/platform/technologies/ws-reliability.html

[54] http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

[55] http://www-128.ibm.com/developerworks/library/specification/ws-wsilspec/

[56] http://www-128.ibm.com/developerworks/library/specification/ws-polfram/

[57] http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-metadataexchange.pdf

[58] http://www-128.ibm.com/developerworks/webservices/library/ws-esbscen/

[59] http://e-docs.bea.com/alsb/docs25/index.html

[60] http://www.iona.com/products/artix/features.htm

[61] http://www.sonicsoftware.com/products/sonic_esb/index.ssp