FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

Integrácia aplikácií pomocou podnikovej zbernice služieb

Diplomová práca

Vladimír Kočí **2007**

Integrácia aplikácií pomocou podnikovej zbernice služieb

DIPLOMOVÁ PRÁCA

Vladimír Kočí

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Informatika

Školiteľ diplomovej práce
Mgr. Pavol Mederly

BRATISLAVA 2007

Zadanie diplomovej práce

Cieľom práce je nájdenie a analýza relevantných štandardov, technológií a produktov týkajúcich sa integrácie aplikácií prostredníctvom podnikovej zbernice služieb (Enterprise Service Bus, ESB). Analyzované a porovnané budú vybrané produkty s otvoreným zdrojovým kódom (napr. Open ESB) a komerčný produkt Sonic ESB. Práca má zohľadniť možné použitie analyzovaných produktov na integráciu aplikácií na Univerzite Komenského.

Čestne vyhlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry.

V Bratislave 30.4.2007

Vladimír Kočí

Ďakujem svojmu diplomovému vedúcemu Mgr. Pavlovi Mederlymu za odborné vedenie práce, cenné rady a pripomienky.

Abstrakt

Autor:	Vladimír Kočí
Názov práce:	Integrácia aplikácií pomocou podnikovej zbernice služieb
Škola:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Vedúci diplomovej práce:	Mgr. Pavol Mederly
Rok:	2007
Počet strán vrátane príloh:	80

Predložená diplomová práca sa zaoberá problematikou integrácie aplikácií pomocou architektúry označovanej ako podniková zbernica služieb (*Enterprise Service Bus - ESB*). V práci je prezentovaná definícia, charakteristiky a architektúra ESB. Práca prináša analýzu a porovnanie štyroch produktov typu ESB – Sonic ESB, Open ESB, Apache ServiceMix a Mule. Práca obsahuje zhrnutie špecifikácie Java Business Integration (JBI), ktorú implementujú dva z analyzovaných produktov.

Práca využíva informačný systém Univerzity Komenského ako modelový podnikový informačný systém. V práci sa identifikujú kritériá na integrovaný informačný systém UK a tieto kritériá sa využívajú na porovnávanie analyzovaných produktov.

Informácie prezentované v práci slúžia na pochopenie architektúry ESB a princípov, z ktorých vychádza. Informácie získané z porovnávania produktov sa dajú využiť pri rozhodovaní o vhodnosti daných produktov v integračnom scenári.

Kľúčové slová:

Integrácia, Enterprise Service Bus, podniková zbernica služieb, Java Business Integration

Predhovor

Predložená diplomová práca sa venuje problematike integrácie aplikácií pomocou podnikovej zbernice služieb (Enterprise Service Bus - ESB). ESB je relatívne mladá integračná architektúra, ktorá vychádza z reálnych potrieb a zo skúseností získaných pri použití mnohých iných integračných architektúr. V súčasnosti však pojem ESB nie je presne definovaný - pod týmto názvom sa skrýva skôr súbor odporúčaní a dobrých zvyklostí.

Centrum informačných technológií Univerzity Komenského plánovalo v rámci svojej integračnej stratégie na UK použiť niektorý z dostupných produktov typu ESB, konkrétne sa zvažovalo zakúpenie produktu Sonic ESB. Nakoľko však o oblasti ESB nie sú jednoznačné informácie, vznikla potreba práce, ktorá by prehľadne zhrnula črty, výhody a nevýhody architektúry ESB. V rámci práce mal byť produkt Sonic ESB otestovaný a porovnaný s ďalšími dostupnými produktmi.

Vychádzajúc z vyššie uvedených potrieb ponúkame v predloženej práci prehľad architektúry ESB – definujeme tento pojem, uvádzame jeho hlavné črty, výhody a nevýhody. Opisujeme problémy, ktoré sa pri hľadaní definície ESB vyskytli, riešime ich definovaním kľúčových pojmov, ktoré ďalej používame. Na architektúru ESB sa pozeráme na troch úrovniach, ktoré ďalej delíme na podúrovne. Informačný systém Univerzity Komenského uvažujeme ako modelový príklad podnikového informačného systému a stanovujeme kritériá, ktoré má spĺňať integrovaný informačný systém UK, nezávisle od použitej integračnej architektúry.

V práci prinášame analýzu a porovnanie štyroch produktov typu ESB. Komerčný produkt Sonic ESB bol určený v zadaní práce, ostatné tri analyzované produkty - Open ESB, Apache ServiceMix a Mule patria medzi hlavné open source produkty typu ESB v súčasnosti. Open ESB a Apache ServiceMix implementujú špecifikáciu JBI, Mule predstavuje odľahčený správový *framework* postavený na princípoch ESB, preto vhodne dopĺňa ostatné analyzované produkty. Produkty sú analyzované a porovnávané jednak v spoločných črtách a vlastnostiach, ale aj na základe identifikovaných požiadaviek na integrovaný informačný systém UK.

O problematike ESB existuje pomerne málo tlačených zdrojov, primárnym zdrojom tejto práce o ESB je [CHA04], ďalšie tlačené zdroje sa ESB dotýkajú len okrajovo. Tlačené zdroje sú v práci vhodne doplnené o informácie získané z viacerých internetových zdrojov, či už z oblasti ESB alebo príbuzných oblastí. Pri identifikovaní požiadaviek na integrovaný informačný systém UK sme vychádzali najmä z interných materiálov Centra informačných technológií UK, pri analyzovaní produktov sme vychádzali predovšetkým z ich dokumentácie a vlastného testovania.

Nakoľko sa stratégia integrácie informačného systému UK stále tvorí, táto práca môže pomôcť práve v tom, že prináša kritériá na integrovaný informačný systém UK, nezávislé na integračnej architektúre. Práca môže byť prínosom aj pre iné organizácie podobného charakteru - poskytuje relevantné informácie použiteľné pri definovaní integračnej stratégie založenej na ESB, resp. pri výbere vhodného produktu typu ESB.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Prirodzeným vývojom organizácie, či podniku je rast, rozširovanie, prípadne globalizácia. Organizácia sa musí vedieť prispôsobiť iným organizáciám, či trhu. Dôsledkom takéhoto vývoja je aj vývoj informačného systému (IS) organizácie. V minulosti postačovali izolované systémy, kde sa každý systém venoval svojej doméne, napr. účtovníctvu alebo skladovému hospodárstvu. Spolu s rastom organizácie a globalizáciou trhu však prichádza nutnosť jednotlivé IS prepájať – integrovať. Prepájajú sa však nielen systémy v rámci organizácie, ale aj systémy v rôznych spolupracujúcich organizáciách. Vzniká tak akási „rozšírená organizácia" („*extended enterprise*" – [CHA04]), do ktorej patrí organizácia a jej obchodní partneri.

Integrovanie systémov organizácie so sebou prináša viacero výhod, medzi ktoré patrí aj zvýšenie konzistencie a odstránenie redundancie dát, či zrýchlenie odozvy systému. Tieto výhody by priniesol aj vývoj nového informačného systému pre organizáciu, oproti tomuto riešeniu však integrácia prináša zhodnotenie prostriedkov investovaných do IS. Navyše, z rôznych organizačných, obchodných a iných dôvodov nemusí byť možné nahradiť existujúce systémy novými.

[CHA04] alebo [MEDIA06] uvádzajú, že integrácia je rovnako technickým, ako aj organizačným problémom. Po technickej stránke treba riešiť otázky ako sprístupnenie funkcionality jednotlivých systémov pre ostatné systémy, zaistenie spoľahlivej a bezpečnej komunikácie medzi systémami a zaistenie kompatibility prenášaných údajov, čo sa týka ich syntaxe a sémantiky v jednotlivých systémoch. Po organizačnej stránke treba zaistiť najmä to, aby používatelia jednotlivých integrovaných aplikácií rešpektovali fakt, že údaje, ktoré do „svojho" systému vkladajú, majú význam pre všetky aplikácie v informačnom systéme. [MEDIA06]

V nasledujúcej kapitole prinášame zhrnutie hlavných foriem integrácie a integračných architektúr. Jednou z týchto foriem je aj integrácia pomocou podnikovej zbernice služieb (*Enterprise Service Bus - ESB*), ktorá tvorí ťažisko tejto práce. ESB patrí v súčasnosti k veľmi často spomínaným pojmom, napriek tomu medzi dodávateľmi produktov, či riešení založených na ESB existujú nezhody o presnej definícii ESB. Významným krokom k štandardizácii ESB môže byť špecifikácia Java Business Integration (JBI), ktorá prináša štandard pre kontajnerové architektúry založené na komunikácii prostredníctvom výmeny správ. Týmto a mnohým ďalším vlastnostiam, črtám a problémom ESB sa venujeme v nasledujúcich kapitolách.

Práca je členená do nasledovných kapitol:

Kapitola 1 prináša zhrnutie najčastejších foriem a architektúr využívaných pri integrácii systémov.

Kapitola 2 sa venuje integrácii v informačnom systéme Univerzity Komenského a sú v nej zhrnuté identifikované kritériá pre integrovaný informačný systém UK.

Kapitola 3 sa venuje integrácii pomocou ESB, obsahuje definíciu a popis vlastností ESB, ako aj popis kľúčových pojmov architektúry ESB. Taktiež sa v tejto kapitole

nachádza porovnanie ESB s inými integračnými architektúrami a popis špecifikácie Java Business Integration.

Kapitola 4 sa venuje analýze a porovnaniu produktov Sonic ESB, Open ESB, Apache ServiceMix a Mule.

Práca je doplnená o rozsiahly slovník pojmov a skratiek, v ktorom sú zhrnuté pojmy, skratky, názvy produktov a technológií a taktiež anglické výrazy spolu s prekladom. V texte práce sú anglické výrazy označené *kurzívou*, výnimku tvoria výrazy, ktoré sú podľa nášho názoru dostatočne „udomácnené“.

V závere práce sú prílohy obsahujúce prehľad komerčných a open source produktov typu ESB prevzatý z [FRY06] a zoznam rozhraní jednotlivých systémov v informačnom systéme UK, pomocou ktorých by tieto systémy mohli byť integrované.

1 Integrácia

Ako sme naznačili v úvode práce, integrácia aplikácií je významná, avšak značne komplexná oblasť. V tejto kapitole uvedieme niektoré hlavné formy integrácie, rôzne spôsoby a techniky integrácie a charakterizujeme hlavné integračné architektúry.

1.1 Formy integrácie

Integrácia systémov môže mať rôzne formy. Rozoznávame najmä internú integráciu aplikácií (*Enterprise Application Integration - EAI*)[1] a externú integráciu aplikácií (*Business-to-business integration – B2B*). Úlohou internej integrácie je primárne prepájať systémy v rámci organizácie, úlohou externej integrácie je potom zabezpečiť komunikáciu medzi systémami partnerských organizácií. Prirodzene, tieto dve formy majú odlišné požiadavky, napríklad z pohľadu bezpečnosti môže pri internej integrácii organizácia viac dôverovať svojim vnútorným systémom, ako môže pri externej integrácii dôverovať systémom partnerskej organizácie. My sa v tejto práci venujeme internej forme integrácie aplikácií (EAI).

Integrácia môže prebiehať na viacerých úrovniach – na úrovni dát, na úrovni služieb alebo na úrovni podnikových procesov. Pri integrácii na úrovni dát môžu jednotlivé systémy zdieľať spoločnú databázu, alebo byť prepojené napr. pomocou techniky označovanej ako ETL. ETL pozostáva z vytiahnutia (*extract*) dát zo zdrojového systému, prenosu (*transfer*) dát do cieľového systému a nahratia (*load*) dát do cieľového systému. Pri integrácii na úrovni služieb je funkcionalita jednotlivých systémov sprístupnená vo forme produkovaných služieb, tieto služby môžu potom využívať (konzumovať) iné systémy, resp. služby v nich. Integrácii na úrovni služieb sa venuje oblasť službovo orientovaných architektúr (*Service Oriented Architecture - SOA*), dobrým zdrojom informácií môže byť napríklad [KRA05]. Integrácia na úrovni podnikových procesov má za úlohu prepájať funkcionality poskytované systémami do definovaných procesov, ktoré majú pre podnik význam, napríklad obchodný.

Ďalším možným delením integrácie uvádzaným v [HOH04] je delenie podľa použitých techník. Rozoznávame integráciu pomocou súborov (sem patrí aj technika ETL), integráciu pomocou zdieľanej databázy, pomocou vzdialeného volania procedúr (*Remote Procedure Call - RPC*) alebo pomocou posielania správ. V tejto práci sa venujeme integrácii na úrovni služieb prostredníctvom posielania správ. Detailnejší popis ostatných foriem, či úrovní integrácie sa nachádza napríklad v [PÁL06]. Technike RPC sa hlbšie venujeme v kapitole 3.5.1 RPC - Remote Procedure Call.

Pri posielaní správ si systémy vymieňajú informácie pomocou diskrétnych kúskov dát (správ), ktoré sú prenášané z jedného systému do druhého. O prenášanie sa stará medzivrstva na posielanie správ, označovaná ako MOM (*Message Oriented Middleware*). Správa spravidla obsahuje hlavičku a telo, v hlavičke môže byť uvedená napríklad adresa cieľového systému alebo ďalšie vlastnosti správy.

[1] Viď kapitola Slovník pojmov.

Posielanie správ je spravidla asynchrónne a môže prebiehať v móde bod-bod (*point-to-point*) alebo v móde *publish-subscribe*.

Posielanie v móde bod-bod je realizované pomocou fronty (*queue*) a je spravidla určené na komunikáciu dvoch systémov. Jeden, označovaný ako producent, do fronty správy vkladá a druhý, označovaný ako konzument, ich vyberá. Správy čakajú vo fronte, kým ich konzument nevyberie. Posielanie v móde *publish-subscribe* je určené na komunikáciu viacerých systémov a je realizované pomocou tém (*topic*). Všetky systémy, ktoré chcú prijímať (konzumovať) správy z témy sa do nej zapíšu (*subscribe*). Keď producent, tiež označovaný ako *publisher*, pošle správu do témy, správa je preposlaná všetkým zapísaným systémom a z témy sa zmaže. V prípade, že sú nejakí konzumenti zapísaní pomocou trvalého zápisu (*durable subscription*), správa v téme sa zmaže až keď je poslaná všetkým trvale zapísaným systémom. Takýmto spôsobom môže konzument dostávať správy aj v prípade, že je v čase poslania správy producentom neprítomný.

1.2 Pevné a voľné prepojenia

Pri prepájaní systémov môže ísť o pevné prepojenia (*tight coupling*) alebo voľné prepojenia (*loose coupling*). Toto rozdelenie je založené na tom, koľko predpokladov má jedna strana o druhej.

Podľa [CHA04] je typickým príkladom pevného prepojenia práve RPC, nakoľko komunikujúce systémy potrebujú poznať detailné informácie o svojich rozhraniach - od adresy, na ktorej je druhý systém prístupný, cez signatúry metód až po typy parametrov. Výhodou pevných prepojení je spravidla ľahšia implementácia, napr. volania pomocou RPC sú podobné volaniam lokálnych funkcií či metód. Veľkou nevýhodou pevných prepojení je však fakt, že zmena v implementácii jedného systému má za následok nutnú zmenu v implementácii ďalších systémov. Navyše, organizácia často nemá kontrolu nad zmenou implementácie niektorých systémov, napríklad systémov, ktoré sú komerčne dodávané alebo systémov obchodných partnerov.

Naopak, pri voľných prepojeniach sa snažíme minimalizovať závislosť jednotlivých systémov na sebe. Pri integrácii na úrovni služieb poznajú jednotlivé systémy, resp. služby len svoje rozhrania, nie sú dôležité implementačné detaily jednotlivých systémov. Pri integrácii pomocou správ môžme využiť asynchrónne posielanie a prijímanie správ, nemusíme preto od systémov vyžadovať, aby boli prístupné v rovnakom čase. Jednou z hlavných výhod voľného prepojenia je ľahké prispôsobenie sa zmene, napríklad pri zmene implementácie jedného systému nie je potrebné meniť implementáciu iných systémov. Nevýhodou môže byť pridaná zložitosť pri dizajne, vývoji a ladení takéhoto riešenia. Tieto nevýhody sa dajú prekonať použitím vhodnej integračnej architektúry.

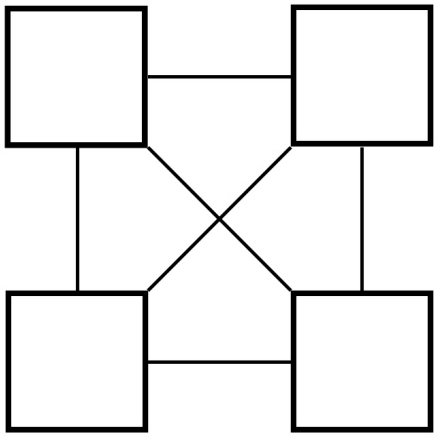
1.3 Integračné architektúry

Hlavnými integračnými architektúrami sú integrácia typu bod-bod (*point-to-point*), integrácia pomocou centrálneho sprostredkovateľa (*broker*) a integrácia pomocou zbernice (*bus*). Podľa [GOEL] sú sprostredkovateľská a zbernicová architektúra dve

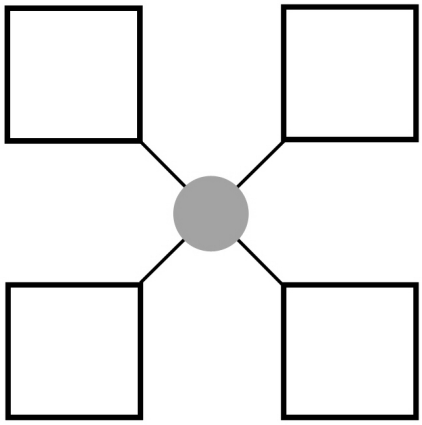
hlavné architektúry, resp. topológie, pomocou ktorých sa realizujú princípy internej formy integrácie aplikácií (EAI).

Pri integrácii typu bod-bod je každý systém prepojený s každým ďalším systémom, ide teda o istý spôsob pevného prepojenia medzi systémami (viď Obrázok 1 - Architektúra bod-bod). Údržba takéhoto integrovaného systému je zložitá, nakoľko v prípade n systémov dostávame analógiu s n-vrcholovým kompletným grafom, kde každá hrana znamená rozhranie medzi dvoma systémami, ktoré treba udržovať.

Integrácia pomocou sprostredkovateľov, tiež označovaná ako *hub-and-spoke* (viď obrázok Obrázok 2 - Architektúra *hub-and-spoke*), využíva centralizovaného sprostredkovateľa (*broker*). Na sprostredkovateľa, ktorý je tiež označovaný ako hub, sa pripájajú aplikácie pomocou adaptérov - *spokes*. Adaptéry sa starajú o transformáciu správ medzi formátom aplikácie a formátom sprostredkovateľa, sprostredkovateľ má za úlohu validáciu, smerovanie a doručovanie správ. Pri použití jedného sprostredkovateľa je ľahká centralizovaná administrácia, takéto riešenie však nie je dobre škálovateľné a jeden sprostredkovateľ môže ľahko znamenať bod zlyhania (*single point of failure*). Federovaná *hub-and-spoke* architektúra rieši tento problém pomocou viacerých sprostredkovateľov, ktorí zdieľajú smerovacie a iné pravidlá (viď Obrázok 3 - Federovaná architektúra *hub-and-spoke*). V prípade výpadku jedného sprostredkovateľa preberajú jeho úlohu ďalší sprostredkovatelia. Federovaná *hub-and-spoke* architektúra sa už čiastočne podobá na zbernicovú architektúru.

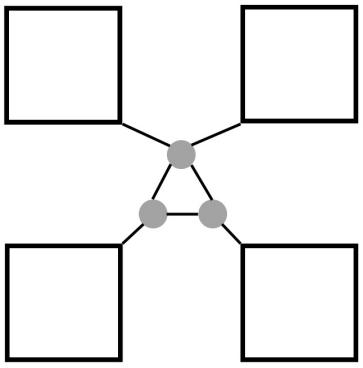


Obrázok 1 - Architektúra bod-bod

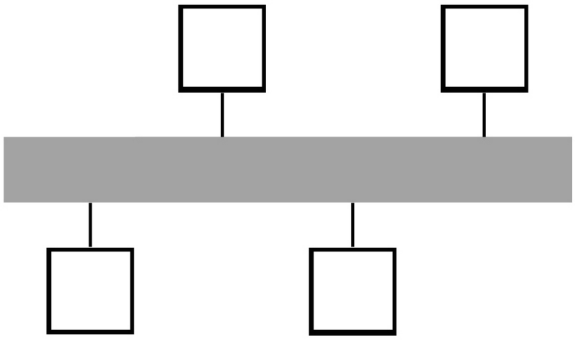


Obrázok 2 - Architektúra *hub-and-spoke*

Zbernicová architektúra využíva spoločnú zbernicu, na ktorú sa pripájajú aplikácie pomocou adaptérov (viď Obrázok 4 - Zbernicová architektúra). Funkcionalita na zbernici je distribuovaná, samotná zbernica môže byť prepojením viacerých zberníc. Táto architektúra je preto škálovateľnejšia ako *hub-and-spoke*, je však ťažšie spravovateľná. Podniková zbernica služieb, ktorej sa venujeme v tejto práci sa radí medzi zbernicové architektúry. Rôznym zbernicovým architektúram, okrem iných aj podnikovej zbernici služieb (ESB), ktorá tvorí ťažisko tejto práce sa venujeme v kapitole 3.2 Hľadanie definície ESB.



Obrázok 3 - Federovaná architektúra *hub-and-spoke*



Obrázok 4 - Zbernicová architektúra

2 Integrácia v informačnom systéme Univerzity Komenského

Informačný systém Univerzity Komenského (ďalej IS UK) pozostáva z približne dvanástich hlavných systémov a aplikácií. Pokrývajú evidenciu študentov, zamestnancov, študentských a zamestnaneckých preukazov, predmetov, rozvrhov, ďalej je tu knižničný systém, ubytovacie systémy na internátoch atď. Ide teda o veľmi rôznorodé aplikácie, často napísané v zastaraných technológiách a použitím zastaraných architektúr. Väčšina z týchto aplikácií pracuje samostatne, tj. nie sú prepojené s ostatnými do formy distribuovaného systému. V dôsledku toho dochádza k redundancii dát (mnohé systémy obsahujú napríklad rovnaké údaje o osobách, ako sú meno a priezvisko), spomaľovaniu viditeľnosti zmien v systémoch (napríklad do terminálov na vstupe do budov alebo učební sa neprenášajú dáta o osobách ihneď keď nastane zmena, ale sú prenášané dávkovo v určitých časových intervaloch) a podobne.

Od roku 2004 prebiehajú na univerzite aktivity smerujúce k prepojeniu týchto systémov, výslednou formou by mal byť integrovaný informačný systém UK (IIS UK)[2], kde budú údaje o objektoch IIS UK kompatibilné medzi jednotlivými systémami, automaticky alebo poloautomaticky prenášané medzi systémami a okrem štandardných používateľských rozhraní budú objekty jednotným spôsobom prístupné pre oprávnených používateľov a oprávnené aplikácie [INT07]. Prínos integračných aktivít je vidno už teraz, aj keď celý proces integrácie je stále vo vývoji. Podarilo sa napríklad zrýchliť proces elektronickej validácie preukazu študenta z niekoľko dní na niekoľko hodín, automatizovať zakladanie záznamov o čitateľoch-študentoch v knižniciach univerzity, odstrániť nutnosť správy používateľov v prístupovom systéme, systéme na správu siete VoIP a ďalších systémoch [MEDIA06].

Hlbšie sa analýzou súčasného stavu IIS UK zaoberajú práce [KOP07], [PÁL06], [SVA07] a [TER06], okrem týchto zdrojov sme vychádzali z [INT06], [INT07], [MEDAR07], [MEDIA06], [VOZ05] a ďalších interných materiálov Centra informačných technológií UK[3].

Táto práca voľne nadväzuje na prácu [TER06], ktorá sa problematike podnikovej zbernice služieb venovala len okrajovo. [TER06] sa sústredila na integráciu systémov IS UK na úrovni dát a to špecificky z pohľadu informácií o osobách. Jedným z výstupov [TER06] bol aj systém CDO (Centrálna Databáza Osôb), ktorý je v súčasnosti prevádzkovaný na UK.

---

[2] Informačný systém Univerzity Komenského má v súčasnosti oficiálny názov Integrovaný informačný a komunikačný systém UK (IIKS UK), z pohľadu tejto práce sa však nejedná o integrovaný systém, preto súčasný systém nazývame iba informačný systém UK (IS UK) a systém, ktorý z pohľadu našej práce bude spĺňať kritériá na integrovaný systém označujeme integrovaný informačný systém UK (IIS UK).

[3] MEDERLY, P. *Integrácia aplikácií – návrh konkrétnych krokov*; MEDERLY, P. *Integrácia aplikácií – základná informácia*; MEDERLY, P. *Integrácia aplikácií – stav k 16.6.2006*

2.1 Kritériá pre integráciu

Účelom tejto kapitoly je stanoviť kritériá, ktoré musí spĺňať integračné riešenie pre IIS UK. V kapitole 4 Analyzované produkty využijeme získané kritériá pre porovnávania produktov ESB „zhora“. Nakoľko sa však konkrétna predstava o integrácii na Univerzite Komenského v súčasnosti len vytvára, tieto kritériá slúžia len na približné vymedzenie hraníc pri porovnávaní.

Tieto kritériá by malo spĺňať ľubovoľné riešenie integrujúce informačný systém Univerzity Komenského do jednotného IIS UK. Tieto výsledky sú teda použiteľné aj pre skúmanie iných možností integrácie, ako len integrácie pomocou ESB.

2.1.1 Nie funkčné požiadavky

- **Rôznorodosť systémov.** Prepojenie cca 12 hlavných systémov a niekoľkých pomocných systémov. Medzi hlavné systémy patria: systém Študent, finančný informačný systém (FIS) vrátane agendy Personalistika a mzdy (PaM), virtuálna knižnica, systém pre správu VoIP (Voice over IP), prístupový systém, univerzitné validačné terminály, softvér pre správu IT prostriedkov, ubytovacie aplikácie, systém realizujúci študentskú anketu, spoločnosť EMcard zaisťujúca funkčnosť preukazov študenta v externom prostredí a Centrálna databáza osôb (CDO). Systémy sú založené na rôznych technológiách, od zastaraných ako napríklad Clipper, DBF (Študent), až po moderné, ako napríklad J2EE, Hibernate (CDO). Prehľad jednotlivých systémov a techológií je v prílohe B: Rozhrania systémov v rámci IS UK, komplexnejšia analýza systémov z pohľadu informácií o osobách sa nachádza v [TER06], z pohľadu študijnej agendy v [SVA07].

- **Rôznorodosť prepojení.** Keďže existujúce systémy sú veľmi rôznorodé, integračné riešenie musí byť schopné podporovať rôzne spôsoby prepojenia systémov. Integrácia niektorých systémov je možná len pomocou dávkových importov a exportov dát (starý systém Študent), iné systémy sprístupňujú svoje dáta na úrovni databáz, pomocou SAP R/3 adaptérov (FIS) alebo sú schopné komunikovať pomocou správ (CDO). Rozsah komunikácie môže byť od malých správ o zmenách až po dávkové importy a exporty celej databázy. Rámcový popis možností komunikácie s existujúcimi systémami je v prílohe B: Rozhrania systémov v rámci IS UK.

- **Odozva.** V integrovanom systéme musia vedieť koncové systémy spolupracovať s rýchlou odozvou. Napríklad pri návrhu systému Študent II (Študent NG) ([VOZ05]) sa počíta s odozvou pod 3 sekundy. V prípade, že bude v IIS UK systém Študent II využívať iné systémy, napríklad Centrálnu databázu osôb, musí byť integračné riešenie schopné poskytnúť požadovanú rýchlosť prepojenia.

- **Dostupnosť.** Dostupnosť systémov je rôzna podľa konkrétneho systému, niektoré tolerujú aj dvojhodinový výpadok v pracovnej dobe, iné musia byť dostupné 24 hodín denne, 7 dní v týždni [VOZ05]. V súčasnosti však na UK neexistuje žiadna IT pohotovosť, tj. výpadky mimo pracovných hodín sa

v momentálnom systéme neriešia. Vďaka všeobecnej „podnikovej“ povahe IIS UK (tj. nejde o časovo kritické aplikácie alebo systémy pracujúce v reálnom čase a podobne) sa dá odhadnúť, že integračné riešenie má tolerovateľný výpadok spolu 10 pracovných hodín za rok. Z hľadiska rozšíriteľnosti do budúcnosti by však mohlo byť vhodné, aby integračné riešenie poskytovalo možnosť napríklad automatickej e-mailovej či SMS notifikácie v prípade výpadku systému.

- **Bezpečnosť.** Mnohé zo systémov pracujú s citlivými osobnými údajmi ako sú mená a adresy osôb, rodné čísla a podobne. Zákon 428/2002 Z.z. ukladá za povinnosť ochranu osobných údajov. Aj na základe toho je potrebné, aby IIS UK riešil bezpečnosť, napríklad šifrovanie komunikácie medzi systémami, autentifikáciu oprávnených užívateľov, s plánovanou podporou centrálnej autentifikačnej autority a autorizáciu (používatelia a iné systémy môžu k systémom pristupovať v rôznych roliach). Jednotnému autentifikačnému systému na Univerzite Komenského sa hlbšie venuje práca [KOP07].

  IIS UK musí taktiež podporovať zaznamenávanie (*logging*) výnimočných stavov (chýb) a dôležitých udalostí (napríklad zmena citlivých údajov), s tým súvisiaci audit dát a tiež je dôležitá podpora automatického zálohovania dát.

- **Robustnosť.** IIS UK musí vedieť zvládnuť robustnosť dát v prepojených systémoch: na univerzite sa eviduje 30 tisíc študentov, 13 fakúlt, 5 tisíc zamestnancov, do 15 tisíc predmetov, 25 tisíc prihlášok ročne. Vybrané historické údaje musia byť zo zákona uchovávané 10 rokov.

- **Súčasné prístupy.** V špičke, napríklad počas zápisu, môže k IIS UK pristupovať až 15 tisíc používateľov súčasne.

2.1.2 Požiadavky na vývoj

- **Modulárnosť a flexibilita.** Riešenie integrujúce systémy by malo byť modulárne, podporovať ľahké pridávanie nových modulov a modifikáciu existujúcich. Riešenie by malo byť flexibilné k zmenám v jednotlivých aplikáciách a zmenám v spôsobe ich prepojenia.

2.1.3 Iné požiadavky

- **Univerzálne číselníky.** IIS UK by mal vedieť podporovať prácu s univerzálnymi číselníkmi, ako sú číselník vysokých škôl SR, číselník krajov a okresov SR, PSČ, národností a podobne. S týmito číselníkmi pracuje viacero systémov, číselníky by preto mali byť spravované na jednom mieste aby si ich každá aplikácia nemusela spravovať v sebe.

3 Integrácia pomocou ESB

Jednou z mnohých integračných architektúr je aj architektúra podnikovej zbernice služieb (*Enterprise Service Bus – ESB*). V tejto kapitole priblížime hlavné črty architektúry ESB a na základe týchto čŕt sa pokúsime nájsť jej definíciu, ukážeme ako architektúra ESB vznikla a porovnáme ju s niektorými ďalšími hlavnými integračnými architektúrami.

Ako už názov napovedá, ESB je architektúra orientovaná na služby, patrí teda medzi architektúry realizujúce koncept SOA (*Service Oriented Architecture*). Dobrým zdrojom informácií o SOA môže byť [KRA05] alebo [MAR06]. Hlavnou črtou architektúr s konceptom SOA je použitie voľne prepojených služieb ako kľúčových artefaktov. Pojmu „služba“ sa hlbšie venujeme v nasledujúcej kapitole.

3.1 Služba

Pri architektúre ESB, ako aj mnohých ďalších architektúrach realizujúcich koncept SOA, je „služba“ kľúčový pojem, mnohé zdroje ho však presne nedefinujú. Napríklad v [HOH04] je služba definovaná ako „dobre definovaná funkcia, ktorá je univerzálne dostupná a odpovedá na požiadavky od konzumentov služby“.

[MAR06] definuje SOA a hlavný artefakt - službu - nasledovne:

> „SOA je konceptuálna biznis architektúra (resp. podniková architektúra)[4], kde funkcionalita (aplikačná logika) je sprístupnená používateľom (konzumentom) SOA ako zdieľané opakovane použiteľné služby na sieti IT. Služby v SOA sú moduly aplikačnej funkcionality s vystavenými rozhraniami a sú volané prostredníctvom správ.“

Navyše zdôrazňuje úlohu modelu založeného na službách pri opakovanej použiteľnosti, interoperabilite a integrácii cez všetky procesy a integračné platformy.

Presnejšiu definíciu môžeme nájsť v [KRA05], kde sa definuje služba dvoma spôsobmi.

Prvý spôsob je všeobecná definícia služby:

> Služba je zmysluplná aktivita, ktorú počítačový program vykonáva na základe požiadavky od iného počítačového programu. Služba je teda samostatne pracujúci aplikačný modul, ktorý je prístupný na diaľku. Aplikačné *frontendy* sprístupňujú služby používateľom.

---

[4] Poznámka k slovu „biznis“: Vo väčšine zdrojov sa slovo biznis, resp. jeho anglický ekvivalent *business*, používa vo veľmi hojnom počte, môžme sa stretnúť s pojmami ako *business architecture* alebo *business functionality*. Podľa kontextu, by sa dalo slovíčko *business* preložiť buď ako biznis, alebo ako podnikový (napr. *business process* – podnikový proces). Aby sme sa vyhli zbytočným nedorozumeniam, rozhodli sme používanie slovíčka *business* obmedziť na minimum. Dúfame, že tým neutrpí výpovedná hodnota citácií.

Služby môžu vykonávať aplikačnú logiku súvisiacu s nejakou aplikačnou funkcionalitou, napríklad rezerváciu letenky, alebo môže ísť o prevažne technickú funkcionalitu, napríklad služba môže mať za úlohu ošetrenie transakcií.

Druhý spôsob uvedený v [KRA05] prezentuje hlbší pohľad na službu a uvádza nasledovné časti, ktoré obsahuje služba:

1. **Zmluva.** Neformálny popis služby, obsahujúci zmysel služby, funkcionalitu, obmedzenia a použitie služby. Forma popisu môže byť odlišná podľa typu služby. Nepovinne môže zmluva obsahovať aj formálny popis rozhraní v jazyku IDL, WSDL a pod.

2. **Rozhrania.** Funkcionalita služieb je sprístupnená klientom (aplikačným *frontendom* a iným službám) cez sieť pomocou rozhraní. Popis rozhraní je súčasťou zmluvy, fyzická implementácia rozhraní pozostáva zo *stubov* pre služby, na základe ktorých sa technicky realizuje napojenie na služby.

3. **Implementácia.** Technická realizácia, ktorá napĺňa zmluvu. Pozostáva z jedného alebo viacerých artefaktov ako sú programy, konfiguračné dáta, databázy a podobne. Implementácia sa dá teda ďalej deliť na aplikačnú logiku a dáta.

Podľa [KRA05] je pri podnikových riešeniach SOA často úlohou identifikovať existujúce aplikačné moduly a komponenty a spojiť ich do služieb s vhodnou granularitou. Služby môžu takto byť lepšie zdokumentované a ľahšie použiteľné, ako ich zložky.

Môže sa zdať, že koncept služby je rovnaký ako koncept komponentu, nakoľko oba pojmy predstavujú nejakú časť systému, ktorá poskytuje definované služby a je schopná komunikovať s inými podobnými časťami systému (službami, resp. komponentmi). Aj keď sú tieto dva koncepty podobné, hlavný rozdiel medzi službou a komponentom je v tom, že koncept komponentu sa týka vývoja (implementácie) a nasadzovania nejakej formy kompilovaného kódu, napríklad triedy Java alebo COM. Koncept služby sa týka skôr nasadzovania a behu (*runtime*).

Implementácia služby môže pozostávať z viacerých spolupracujúcich tried, komponentov alebo celých systémov, pričom tieto nemusia byť napísané v rovnakom programovacom jazyku, môžu bežať v rôznych prostrediach a spolupracovať spolu na dodaní funkcionality, ktorú služba poskytuje.

3.2 Hľadanie definície ESB

ESB by sa jednou vetou dalo charakterizovať ako „Voľne prepojená, distribuovaná, udalosťami riadená architektúra orientovaná na služby s dôrazom na konfigurovateľnosť.“ V tejto kapitole definujeme podnikovú zbernicu služieb (ESB) na základe definícií zbernice a zbernice služieb. Taktiež ukážeme, v čom sa vyskytli pri hľadaní definícií problémy.

ESB je relatívne nový architektonický pohľad ([CHA04] považuje za vznik ESB rok 2001), ktorý odmieta „tradičné“ prístupy bod-bod (*point-to-point*) a *hub-and-spoke* architektúr a uprednostňuje voľne prepojenú vysoko distribuovanú sieť.

Nejde sa však o presne špecifikovanú architektúru, za ktorou by stála nejaká konkrétna organizácia, ako tomu je napríklad pri architektúre CORBA od Object Management Group. ESB sa chápe skôr ako súbor odporúčaní, architektonických riešení a dobrých zvyklostí získaných skúsenosťami z iných integračných architektúr.

Keďže nejde o presnú špecifikáciu, rôzni dodávatelia ponúkajú verzie ESB založené na rôznych definíciách a interpretáciách, ktoré spravidla uprednostňujú ich vlastné technológie. Príklad môžeme nájsť na stránke spoločnosti CapeClear [CAPMES], kde sa kriticky porovnávajú „správovo založené ESB“ od konkurencie s vlastným „službovo založeným ESB“.

Článkom, ktorý sa pokúša vysvetliť jednotlivé pojmy a zasadiť ESB do kontextu Enterpise Application Integration (EAI) a SOA môže byť [GOEL]. Definuje EAI ako potrebu na prepojenie rôznych aplikácií v rámci podniku a partnerských systémov aby dosiahli cieľ jednotným a celistvým spôsobom, nezávisle na platforme a geografickej polohe jednotlivých aplikácií. Toto zahŕňa prijímanie, transformáciu, smerovanie a doručovanie správ a manažment procesov. EAI sa dá realizovať dvomi hlavnými architektúrami: *hub-and-spoke* a zbernicovou architektúrou.

Pri hľadaní definície ESB sme narazili na problém nejasnej terminológie - v zdrojoch sa často miešajú pojmy „zbernicová architektúra“ a „zbernicová architektúra založená na službách“, resp. „zbernica služieb“. Tieto pojmy však nie sú ekvivalentné. Ako ukážeme v nasledujúcich kapitolách 3.2.1 Zbernica a 3.2.2 Zbernica služieb, zbernicová architektúra ešte nemusí byť nutne založená na službách. Konfliktné bývajú navyše samotné označenia toho, čo ešte je, a čo už nie je zbernica. Podľa [KRA05] sa za zbernicu považuje aj napríklad CORBA Object Request Broker, zatiaľ čo podľa iných zdrojov je CORBA typickým príkladom architektúry bod-bod (napr. [CHA04]). Podobne, za zbernicu sa podľa niektorých zdrojov ešte nepovažuje federovaná *hub-and-spoke* architektúra (napr. [GOEL]), iné zdroje ju však už za zbernicu považujú, aj keď nemusia nutne považovať za zbernicu služieb (napr. [BAK05]).

Jadrom problému sa zdá byť definícia samotnej zbernice, resp. zbernice služieb. V nasledujúcich kapitolách si zadefinujeme tieto pojmy a uvedieme aj ďalšie možné definície z iných zdrojov. V kapitole 3.2.3 Podniková zbernica služieb (ESB) potom vychádzajúc z týchto definícií definujeme a charakterizujeme samotnú podnikovú zbernicu služieb (ESB).

3.2.1 Zbernica

Keď uvažujeme analógiu s hardvérovou zbernicou alebo napríklad so zbernicou v sieťovej topológii, môžeme zbernicu definovať takto:

a) Zbernica je systém, ktorý sprostredkováva informácie medzi ostatnými systémami, ktoré sú naňho pripojené. Komunikácia medzi koncovými systémami neprebieha priamo, ale cez „zbernicový systém“, teda nie sú

potrebné priame prepojenia medzi koncovými systémami. Koncové systémy nemusia pochádzať „od toho istého dodávateľa“ (ak použijeme pojem z hardvérovej oblasti) - teda môžu byť napríklad napísané v rozdielnych jazykoch. Preto musí zbernica zabezpečiť nejaký štandardný spôsob prepojenia medzi koncovými systémami.

V [KRA05] takto definovanej zbernici zodpovedá pojem *software bus*.

Tejto definícii zbernice by vyhovoval aj napríklad EAI hub, ktorý, ako uvedieme v kapitole 3.5.3 EAI hub-and-spoke, sa radí medzi centralizované architektúry. Definícia je teda pre naše účely príliš všeobecná. Zmeníme teda definíciu zbernice nasledovným spôsobom:

b) Zbernicu definujeme ako bode a) a navyše budeme požadovať, aby zbernicový systém podporoval distribuovanosť – zbernicový systém teda bude pozostávať z jedného alebo viacerých rovnocenných spolupracujúcich subsystémov. Koncové systémy sa nemusia nutne pripájať na ten istý subsystém, avšak systém zbernice zabezpečí komunikáciu v zmysle bode a) – zbernica je teda pre koncové systémy atomická.

Definícia a) je teda špeciálnym prípadom tejto definície, kde zbernicový systém je tvorený práve jedným subsystémom. K zberniciam v zmysle definície b) radíme aj federované *hub-and-spoke* architektúry.

3.2.2 Zbernica služieb

Nad definíciou b) z predchádzajúcej kapitoly môžme postaviť definíciu zbernice služieb, ktorú neskôr použijeme pri charakterizovaní podnikovej zbernice služieb (ESB) v kapitole 3.2.3 Podniková zbernica služieb (ESB):

> Zbernica služieb je zbernicová architektúra, kde zbernica je uvažovaná v zmysle definície b) z kapitoly 3.2.1 Zbernica, ktorá pod koncovými systémami chápe služby v zmysle kapitoly 3.1 Služba.

Sú aj možné aj odlišné definície zbernice služieb, ktoré sa na zbernicu služieb pozerajú ako na meta-zbernicu alebo integračný vzor. Uvádzame tu aj tieto definície, lebo môžu byť nápomocné v pochopení charakteristík podnikovej zbernice služieb (ESB).

Zbernica služieb ako meta-zbernica

[KRA05] pod pojmom *service bus* rozumie akúsi meta-zbernicu, ktorá zastrešuje všetky zbernice (*software bus*) používané v podniku na prepojenie systémov. [KRA05] pod zbernicou rozumie zbernicu v zmysle našej definície a). V prípade tejto zbernice služieb (*service bus*) teda nejde o jednu technológiu, ale o spojenie viacerých technológií (zberníc). Vychádza sa pritom zo skúsenosti, že produkty, ktoré sa v minulosti snažili vyriešiť integráciu jednotným spôsobom väčšinou neboli tak úspešné, ako sa predpokladalo (napr. CORBA alebo konkrétne EAI huby) a preto [KRA05] tvrdí, že je výhodnejšie založiť SOA na meta-zbernici, ktorá bude,

takpovediac, integrovať existujúce integračné riešenia. Medzi hlavné znaky takejto meta-zbernice radí:

- **Prepojenie.** Hlavným cieľom meta-zbernice je prepojiť účastníkov SOA. Meta-zbernica poskytuje prostriedky pre *frontend* aplikácie a služby na vyvolávanie funkcionality iných služieb.

- **Rôznorodosť technológií.** Meta-zbernica musí obsiahnuť a prepájať široké spektrum rozličných technológií, ktoré sa vyskytujú v reálnom podnikovom prostredí. Účastnícke aplikácie môžu byť napísané v rôznych programovacích jazykoch, pracovať pod rôznymi operačnými systémami a *runtime* prostrediami. Taktiež musí meta-zbernica vedieť spolupracovať s veľkým množstvom *middleware* technológií, komunikačných protokolov a podobne.

- **Rôznorodosť komunikačných konceptov.** Podobne ako pri rôznorodosti technológií musí vedieť meta-zbernica taktiež podporovať široké spektrum rozličných komunikačných konceptov a módov, ako sú synchrónna či asynchrónna komunikácia a podobne.

- **Technické služby.** Okrem hlavnej funkcie meta-zbernice, t.j. prepájania systémov, musí táto poskytovať aj „služby“ ako zaznamenávanie (*logging*), audit, bezpečnosť, transformácie správ, transakcie a podobne.

Pri koncepte meta-zbernice je dôraz na zachovanie existujúcich integračných riešení a ich prepojenie novým spôsobom. Ako však ukážeme v kapitole 3.3 Vývoj ESB, pri ESB je skôr dôraz na postupné nahrádzanie nevhodných existujúcich integračných riešení jedným novým spoločným riešením.

Zbernica služieb ako integračný vzor

Iný uhol pohľadu prezentuje napríklad [MAR06] alebo [HOH04], keď sa na zbernicu služieb pozerá ako na integračný vzor, slúžiaci na transport správ. [MAR06] dáva zbernicu do kontrastu s ďalšími dvoma vzormi pre transport - priamym prístupom, kedy služby komunikujú priamo medzi sebou bez nejakého sprostredkovateľa, a so sprostredkovateľským (*broker*) vzorom, ktorý na komunikáciu medzi službami využíva jeden alebo niekoľko sprostredkovateľov.

Pri priamom prístupe je nutné, aby každý producent a konzument správ sami ošetrovali procesné pravidlá, smerovanie správ, bezpečnosť a podobne, preto je táto architektúra, ktorá využíva pevné spojenia skôr vhodná pre homogénne prostredia a integračné požiadavky malého rozsahu.

Pri použití sprostredkovateľov sa spoločná funkcionalita, ako napríklad bezpečnosť alebo smerovanie správ presúva na sprostredkovateľa, resp. sprostredkovateľov, čím sa dosiahne voľné prepojenie producentov a konzumentov správ. Navyše na úrovni sprostredkovateľov sa dá riešiť vyvažovanie záťaže (*load balancing*), transformácie správ, implementácia spoločnej aplikačnej logiky a podobne. Typickou implementáciou sprostredkovateľského vzoru môže byť EAI hub.

Zbernica služieb v zmysle [MAR06] ide ešte o krok ďalej. Od zbernice je očakávané nielen spoľahlivé doručenie správ, *queuing* alebo *publish-subscribe* mód, prípadne dlhotrvajúce transakcie a podobne, čo sú rovnako črty sprostredkovateľských architektúr. Taktiež očakáva riešenie komplexných časovacích (*scheduling*) scenárov alebo procesných tokov, služby môžu byť od začiatku navrhované bez zaťažovania sa problémami komunikácie, smerovaním správ a podobne. Ako uvidíme v kapitole 3.2.3 Podniková zbernica služieb (ESB), podobne chápu zbernicu služieb aj iné zdroje, napr. [CHA04], kde sa vyzdvihuje distribuovaná povaha službových kontajnerov, ktoré priamo kontrastujú s centralizovanou a monolitickou architektúrou sprostredkovateľských architektúr.

3.2.3 Podniková zbernica služieb (ESB)

Vychádzajúc z našej definície zbernice služieb zo začiatku predchádzajúcej kapitoly, môžme definovať podnikovú zbernicu služieb nasledovne:

> Podniková zbernica služieb (ESB) je zbernicová architektúra v zmysle našej definície zbernice služieb zo začiatku kapitoly 3.2.2 Zbernica služieb, ktorej hlavnými charakteristikami sú distribuovanosť, sprostredkované voľné prepojenie medzi službami, spoľahlivá komunikácia prostredníctvom správ, podpora otvorených štandardov a dôraz na konfigurovateľnosť namiesto písania kódu.

Nie je však ľahké podať presnú definíciu ESB. Ako sme spomenuli v úvode kapitoly 3.2 Hľadanie definície ESB, ESB je relatívne nový pojem, preto ešte neexistuje dostatočné množstvo publikácií venujúcich sa definícii tohto pojmu - prakticky jediná tlačená publikácia je [CHA04]. ESB je však v súčasnosti veľmi živá oblasť, preto sa našťastie dá čerpať z internetových zdrojov. Uvádzame spoločné charakteristiky, na ktorých sa zhodne viacero zdrojov, či už tlačených ([CHA04]) alebo internetových. Medzi internetové zdroje patria zväčša zdroje od spoločností vyvíjajúcich produkty typu ESB, napr. [BAK05], [SONDOC], [SMIX] alebo [VDO06].

Okrem spoločných charakteristík uvádzame aj charakteristiky uvedené len v niektorých zdrojoch. Pri každej charakteristike ukážeme podobnosti s inými integračnými architektúrami, tieto potom hlbšie rozvádzame v kapitole 3.5 Porovnanie ESB s inými integračnými architektúrami.

Medzi spoločné charakteristiky patrí:

- **Spoľahlivá komunikácia prostredníctvom správ.** Služby navzájom komunikujú prostredníctvom posielania správ. Toto posielanie môže prebiehať v rôznych režimoch: synchrónne, asynchrónne, bod-bod aj *publish-subscribe*. Správy môžu byť perzistentné, tj. trvalo uchovávané pre prípad, že cieľová služba nie je ihneď dostupná. Toto sa dá realizovať napríklad pomocou techniky *durable subscriptions*, o ktorej sme hovorili v kapitole 1.1 Formy integrácie. Vidieť tu podobnosť s architektúrami typu MOM.

- **Sprostredkovanie (mediácia).** ESB je medzivrstvou medzi poskytovateľmi (producentmi) a prijímateľmi (konzumentmi) služieb, čím podporuje ich voľné prepojenie bez nutnosti vzájomného poznania medzi aplikáciami. Vidno

tu podobnosť s väčšinou integračných architektúr (MOM, WS, EAI). ESB teda musí riešiť okrem iných:

- Adresovanie služieb jednotným a univerzálnym spôsobom, napríklad pomocou koncových bodov.

- Smerovanie správ na základe nejakej centrálnej smerovacej politiky alebo na základe obsahu správ (*Content Based Routing - CBR*). Pod centrálnou politikou sa rozumie napríklad orchestrovaný procesný tok, ktorému sa hlbšie venujeme v kapitole 3.4.3 Úroveň procesov. Pri CBR nie je nutné centrálne riadenie, ale vyžaduje sa znalosť metadát o správach, napríklad informácie o štruktúre tela správy, aby bolo možné definovať smerovacie pravidlá.

- Transformácia správ medzi rôznymi formátmi zdrojových a cieľových aplikácií, napr. pri použití XML správ môže transformácia prebiehať pomocou XSLT. Znova nie je nevyhnutná centralizovaná transformačná politika. Transformácia správ vyžaduje od architekta znalosti metadát o službách, napríklad rozhrania služieb a s tým súvisiaci formát správ, ktorý produkujú či konzumujú.

- **Nezávislé na operačnom systéme a programovacom jazyku aplikácií.** Napríklad umožňuje komunikáciu medzi aplikáciami písanými v jazykoch Java a .NET. Podobné charakteristiky majú aj webové služby.

- **Využitie XML ako formátu správ.** Aj keď toto nie je nutná podmienka, napríklad z dôvodu výkonu môžu byť správy v rýchlejšie spracovateľnom formáte, formát XML ma mnoho výhod: je nezávislý na prostredí, má širokú podporu v aplikáciách a programovacích jazykoch, je transparentný – pri administrácii a monitorovaní je obsah XML správy viditeľný, atď. Výhody XML sa ukázali už pri webových službách.

- **Založené na štandardoch.** Úlohou je obmedziť vlastné technológie na minimum a uprednostniť otvorené štandardy. Komunikácia prostredníctvom správ: HTTP, JMS; prepájanie serverov podnikových systémov: J2EE Connector Architecture; monitorovanie a administrácia: JMX; jazyky v ktorých sú písané služby: Java, C/C++, C#; platformy: J2EE, .NET, COM; správy, transformácie: SOAP, XML, XSLT, XPath, XQuery; popis služieb: WSDL; popis procesov: BPEL4WS; a mnohé ďalšie ako napríklad protokoly sady WS-*. Odlišné ESB produkty však často používajú odlišné štandardy, napríklad Apache ServiceMix a OpenESB používajú ako procesný jazyk BPEL, Sonic ESB používa vlastný štandard podobný diagramom aktivít jazyka UML. Na niekoľkých z týchto otvorených štandardov boli založené už webové služby, staršie integračné architektúry ako napríklad EAI huby väčšinou využívali vlastné štandardy a protokoly, lebo v čase ich vzniku ešte neboli dostupné vhodné otvorené štandardy.

- **Podpora pre orchestráciu služieb.** Komunikácia medzi službami môže byť súčasťou procesného toku a je riadená orchestračným, resp. procesným prostredím. Toto môže napríklad byť implementované ako služba, ktorá sa

pripája na komunikačnú zbernicu a riadi tok správ a volania medzi ostatnými službami. Procesný tok môže byť dlhotrvajúci a stavový, nejde teda len o samotné riadenie komunikácie. Orchestráciu riešili už EAI huby, aj keď pri nich sa jednalo o viac centralizované riešenia. Orchestrácia v ESB nemusí byť nutne centralizovaná, procesný tok môže riadiť viacero prepojených orchestračných služieb.

- **Bezpečnostný model na autorizáciu, autentifikáciu a audit používania komunikačnej zbernice.** Na prepojeniach medzi uzlami na ESB môžu byť umiestnené firewally bez vplyvu na funkčnosť ESB, ESB teda môže zasahovať do viacerých sieťových segmentov - toto môže do veľkej miery umožnené používaním transparentných formátov (HTTP, XML) na rozdiel od vlastných formátov, ako napríklad CORBA IIOP. Je otázne, dokedy bude komunikácia napríklad prostredníctvom formátu HTTP považovaná za neškodnú a nebude sa blokovať na firewalloch – zlomom môže byť napríklad objavenie sa prvého SOAP vírusu. Je však pravdepodobné, že protokoly ako WS-Security budú v budúcnosti riešiť takéto situácie. V prípade, že sa konkrétny produkt ESB opiera na úrovni správ o iný existujúci produkt typu MOM, môže sa bezpečnostný model ESB v mnohých veciach opierať o bezpečnostný model používanej vrstvy MOM. Takto tomu je napríklad pri testovaných produktoch Sonic ESB, Open ESB alebo Apache ServiceMix.

- **Distribuované a postupné nasadzovanie.** Aplikácie môžu byť na zbernicu pripájané postupne, rovnako ako infraštruktúra samotnej zbernice môže byť vytváraná postupne. Pri integrácii je toto zvlášť výhodné, lebo časť integrovaného systému môže pracovať na základe existujúceho prepojenia, zatiaľ čo iná časť systému už môže byť prepojená pomocou ESB. Odpadá teda nutnosť integrácie systému spôsobom „celý systém alebo nič“. Pri iných integračných architektúrach môže byť obmedzená možnosť distribuovaného nasadzovania resp. distribuovaného vytvárania infraštruktúry, napríklad pri centralizovanej architektúre typu EAI *hub-and-spoke* je nutné budovať infraštruktúru okolo centralizovaného sprostredkovateľa.

- **Škálovateľnosť a architektúra orientovaná na dostupnosť.** ESB musí vedieť garantovať rýchlosť a dostupnosť potrebnú pre výmenu veľkého množstva správ medzi aplikáciami či službami. Škálovateľnosť do istej miery súvisí s distribuovanosťou architektúry ESB, viac v kapitole 3.4.2 Úroveň služieb. Škálovateľnosť rieši viacero architektúr, pri ESB je však vďaka vlastnostiam distribuovaných odľahčených kontajnerov škálovateľnosť efektívnejšia.

- **Podpora webových služieb.** Protokoly webových služieb ako XML, WSDL, SOAP, či novšie protokoly WS-* sú podporované v ESB, niektoré zdroje dokonca uvádzajú webové služby ako základ, na ktorom je postavená architektúra ESB ([BAK05]).

- **Infraštruktúra na konfiguráciu, nasadzovanie, riadenie a monitorovanie služieb.** Pri ESB je veľký dôraz na dynamickú konfigurovateľnosť a správu, namiesto písania a kompilovania kódu. Pri použití vhodných štandardov, ako napríklad JMX (Java Management eXtensions) je samozrejmá aj diaľková

správa a konfigurovanie, ktorých výhodu špeciálne vyzdvihuje [CHA04]. Treba poznamenať, že diaľková správa je možná pri viacerých integračných architektúrach, napríklad vďaka centralizovanej architektúre EAI hubov je ich správa značne uľahčená.

Okrem týchto charakteristík považujú niektoré zdroje za dôležité ešte nasledovné charakteristiky:

- **Adresárové služby.** Vyhľadávanie implementácií služieb počas behu [VDO06].

- **Širokosiahľosť.** Podľa [CHA04] má byť ESB schopné byť jadrom heterogénnej siete, ktorá má globálny dosah cez organizačné jednotky, obchodných partnerov a pod. Podobne ako bezpečnostný model, aj širokosiahlosť súvisí s používaním transparentných formátov, ktoré umožňujú zasahovať ESB do viacerých sieťových segmentov, keďže nespôsobujú problémy firewallom.

- **Kanonický dátový model.** Výhody kanonického dátového modelu, resp. formátu dát, vyzdvihuje [HOH04] aj [CHA04]. Správy prechádzajú zbernicou v definovanom formáte a štruktúre, preklad z kanonického dátového modelu a do kanonického dátového modelu sa uskutočňuje pomocou aplikačných adaptérov. V [CHA04] síce kanonický dátový formát nie je povinným, ale veľmi dobrým riešením. Pozitívum vidí jednak v zrejmom sprehľadnení a zjednodušení transformácií správ, ale aj v adaptovaní štandardov ako kanonických formátov, napr. formát adresy, faktúry alebo obchodnej objednávky. Príkladom takéhoto štandardu môže byť xCBL od CommerceOne[5], ktorý obsahuje cez 40 dokumentov pre *business-to-business* komunikáciu. Dvojitá transformácia z formátu zdrojovej aplikácie do kanonického formátu a následná transformácia z kanonického formátu do formátu cieľovej aplikácie má aj svoje nevýhody, keďže predstavuje isté zdržanie. Toto môže byť pri použití „veľkých“ formátov ako XML neprípustné pre časovo kritické aplikácie. Ide teda o zvolenie kompromisu – riešenie s lepšou udržovateľnosťou alebo riešenie s lepšou výkonnosťou.

Niektoré zdroje rozumejú pod ESB zbernicu v zmysle definície a) z kapitoly 3.2.1 Zbernica a zbernicu v zmysle definície b) nazývajú „Distributed ESB“ alebo „Enterprise Service Network“ ([HAR06]).

V [HOH04] sa považuje za základ SOA „správová zbernica“ (*message bus*), ktorá má niektoré črty podobné s ESB. Správová zbernica pracuje so správami v kanonickom formáte, každá služba aspoň jeden kanál na prijímanie požiadaviek a taktiež môže mať jeden alebo viac kanálov na posielanie odpovedí. Posielací kanál z jednej služby je prepojený s prijímacím kanálom inej služby a množina kanálov tak akoby tvorí samotnú zbernicu, ktorá slúži ako základ multiplatformovej jazykovo nezávislej komunikačnej infraštruktúry. Táto infraštruktúra poskytuje služby smerovania správ,

[5] http://www.xcbl.org

posielania viacerým príjemcom (napr. v móde *publish-subscribe*) a podobne. Dôležitou črtou pri správovej zbernici je aj spoločná štandardná štruktúra rozhraní služieb.

3.3 Vývoj ESB

V tejto kapitole ukážeme, že architektúra ESB nevznikla náhodne, ale vyvinula sa na základe skúseností získaných z ostatných, skôr vzniknutých integračných architektúr. Aj keď architektúra ESB nie je nutne naviazaná na integráciu - je výhodná aj pri vývoji nových aplikácií – v ďalšom budeme o ESB uvažovať primárne ako o integračnej architektúre. Budeme pritom vychádzať hlavne z [CHA04], kde sa s ESB pracuje v podobnom kontexte.

[CHA04] sa opiera o poznatky stavu integračných riešení v súčasnosti a ukazuje, ako sa vývojom týchto riešení dospelo až k architektúre ESB. Za hlavné črty súčasných podnikových (*enterprise*) projektov považuje to, že vo všeobecnosti nie sú dostatočne prepojené: podľa citovaných prieskumov spoločnosti Gartner Inc. je menej ako 10% podnikových aplikácií prepojených a z tých, ktoré sú prepojené, len 15% využíva nejakú formálnu alebo štandardizovanú integračnú technológiu[6]. Zvyšok integrácie sa deje prostredníctvom techník ETL (ktoré sme popísali v kapitole 1.1 Formy integrácie), či prostredníctvom dávkového prenosu súborov - príkladom môže byť nočné zhromaždenie údajov, posielanie cez FTP a načítanie týchto údajov v cieľovom systéme. Tieto riešenia majú zrejmé veľké zdržanie (napríklad pri prenose dát v nočných hodinách môže trvať až 24 hodín, kým sa zmena prejaví v cieľovom systéme), problémy so synchronizáciou, zotavením z chýb pri prenose a podobne. Existuje teda akási „náhodná architektúra“ [CHA04], ktorá sa vytvorila časom v dôsledku používania rôznych integračných technológií, bez globálneho plánovania. Ak je aj použitie konkrétnej integračnej technológie dobre zanalyzované a naplánované, pri miešaní viacerých technológií či architektúr postupom času sa len ťažko dá vyhnúť strate dizajnovej integrity.

Všetky tieto skúsenosti ukazujú, že je výhodnejšie zakladať integráciu na otvorených štandardoch (na rozdiel od uzavretých riešení naviazaných na dodávateľov, ako tomu bolo pri tradičných EAI huboch - spomína sa napríklad v [BAK05]), riešeniach podporujúcich postupnú integráciu aplikácií a ľahkú rozšíriteľnosť celej architektúry, pričom integračná sieť by mala umožňovať rozšírenie až za hranice podniku a zasahovať aj do systémov partnerských firiem ([CHA04] používa na popísanie takéhoto podniku pojem *extended enterprise*). Práve ESB má byť vhodným kandidátom na takúto architektúru.

Mnohé z nám dostupných relevantných zdrojov veria, že ESB vzniklo evolúciou z viacerých technológií. [CHA04] považuje za míľnikový rok 2001, kedy sa dostatočná „zrelosť“ štandardov ako XML, JMS, E-komerčných serverov, webových služieb a potreba vybudovať štandardizovanú integračnú sieťovú architektúru pretransformovala do podoby ESB.

---

[6] *Integration Brokers, Application Servers and APSs*, Gartner Inc., štatistika z 10/2002, podľa [CHA04]

ESB čerpá, resp. je inšpirovaná viacerými technológiami, architektúrami či dokonca udalosťami. Podľa [CHA04] sú to nasledovné:

- **Enterprise Application Integration.** EAI huby z konca 90. rokov mali obmedzené možnosti zasahovania mimo fyzické sieťové segmenty, nepoužívali otvorené štandardy a vývoj integračných riešení trval buď príliš dlho (priemer 20 a viac mesiacov) alebo prekročil plánované zdroje (menej ako 35% projektov dokončených načas a s plánovanými financiami) a údržba sa sústredila prevažne na prepojenia bod-bod medzi aplikáciami (35% zdrojov na údržbu)[7]. Z EAI sa prevzali myšlienky transformácie a manipulácie dát, vhodne prispôsobené pre využitie v XML technológiách. O EAI huboch píšeme viac v kapitole 3.5.3 EAI hub-and-spoke.

- **E-komerčné servery.** Vo svojich počiatkoch revolučné myšlienky, ktoré prepájali systémy obchodných partnerov voľným spôsobom a ktoré mali byť univerzálnym riešením a formou obchodu, sa ukázali byť ako príliš nereálne očakávania a prasknutím príliš nafukovanej internetovej bubliny v 90. rokoch zaznamenali pokles. Napriek tomu, koncepty voľne prepojených systémov a architektúr založených na otvorených štandardoch sa zachovali.

- **Java Message Service.** Vďaka JMS ako súčasť platformy J2EE sa produkty typu MOM dostali do „hlavného prúdu“ a vytvoril sa konkurenčný trh dodávateľov správovo orientovaných systémov. MOM sa viac venujeme v kapitole 3.5.2 MOM - Message Oriented Middleware.

- **Aplikačné servery.** Ukázali výhody oddelenia aplikačnej logiky od dát a prezentácie (podľa dizajnového vzoru Model-View-Controller) a pomohli pri vzniku štandardov ako servlety na dynamické spracovanie požiadaviek, JDBC na jednotný prístup k databázam a J2EE Connector Architecture (JCA) na prepojenie aplikačných adaptérov.

- **Y2K a ekonomika po internetovej bubline.** Koncom 90. rokov sa začalo prehodnocovať vyvíjanie rozsiahlych aplikácií vlastnými silami, ale aj investovanie do veľkých softwarových produktov. Začali sa uprednostňovať riešenia, ktoré vhodne prepoja a využijú existujúce systémy a komerčne dostupné „hotové“ systémy prispôsobené požiadavkám konkrétnej firmy.

- **Webové služby a SOA.** Koncepty webových služieb sú dôležitou súčasťou ESB a na samotné ESB patrí medzi architektúry realizujúce princípy SOA.

[CHA04] teda v ESB vidí pokračovanie integračných architektúr z minulosti, ktoré však už nemusia úplne vyhovovať aktuálnym potrebám.

V súčasnej dobe je pojem ESB veľmi často skloňovaný a mnoho firiem sa snaží svoje produkty predať ako ESB, čomu iste napomáha aj fakt, že chýba presná definícia ESB. Príkladom môžu byť tradičné produkty typu EAI hub, ktoré boli na trhu už od

[7] *Reducing Integration Costs*, Forrester Research, štatistika z 12/2001, podľa [CHA04]

90. rokov ako Microsoft BizTalk, TIBCO BusinessWork, IBM WebSphere a mnohé ďalšie, ktoré podporovali architektúru, ktorú sami označovali za zbernicovú. Aj keď tieto produkty spĺňajú mnohé charakteristiky ESB, stále ide skôr o cenovo náročnejšie monolitické riešenia založené na uzavretých štandardoch, v čom niektoré zdroje, napr. [BAK05], [CHA04] alebo [GOEL] vidia ich nevýhodu oproti „pravým“ ESB v zmysle našej definície.

Záujem trhu a IT komunity o ESB je dobre vidieť aj v analýzach firmy Gartner Inc., ktorá každoročne formou grafu Hype Cycle mapuje, okrem iných, aj integračné a middlewarové technológie. Hype Cycle graf dáva do súvislosti zviditeľňovanie sa produktov či technológií s ich vyspelosťou a má 5 fáz: stúpajúca (technológia prišla na trh a začína sa o nej hovoriť), na vrchole očakávaní (technológia je pomerne nová sú na ňu kladené vysoké očakávania), prepad (vytriezvenie z privysokých očakávaní), stúpanie k „osvieteniu“ (zisťujú sa skutočné pozitíva technológie), rovina produktivity (je zrejmá skutočná hodnota a produktivita technológie). Tento graf celkom adekvátne modeluje nástup a uplatňovanie sa nových technológií. Technológia ESB sa presunula zo začiatočnej stúpajúcej pozície v rokoch 2003 [GAR03] a 2004 [GAR04] na vrchol očakávaní v rokoch 2005 [GAR05] a 2006 [GAR06]. Medzi technológie v piatej fáze sa radí napríklad J2EE a MOM, mnohé technológie sa však nedostanú ďalej ako po tretiu fázu alebo takpovediac zmeškajú svoj čas (napríklad ebXML Messaging Specification). Pre ilustráciu - SOA sa radí do tretej fázy.

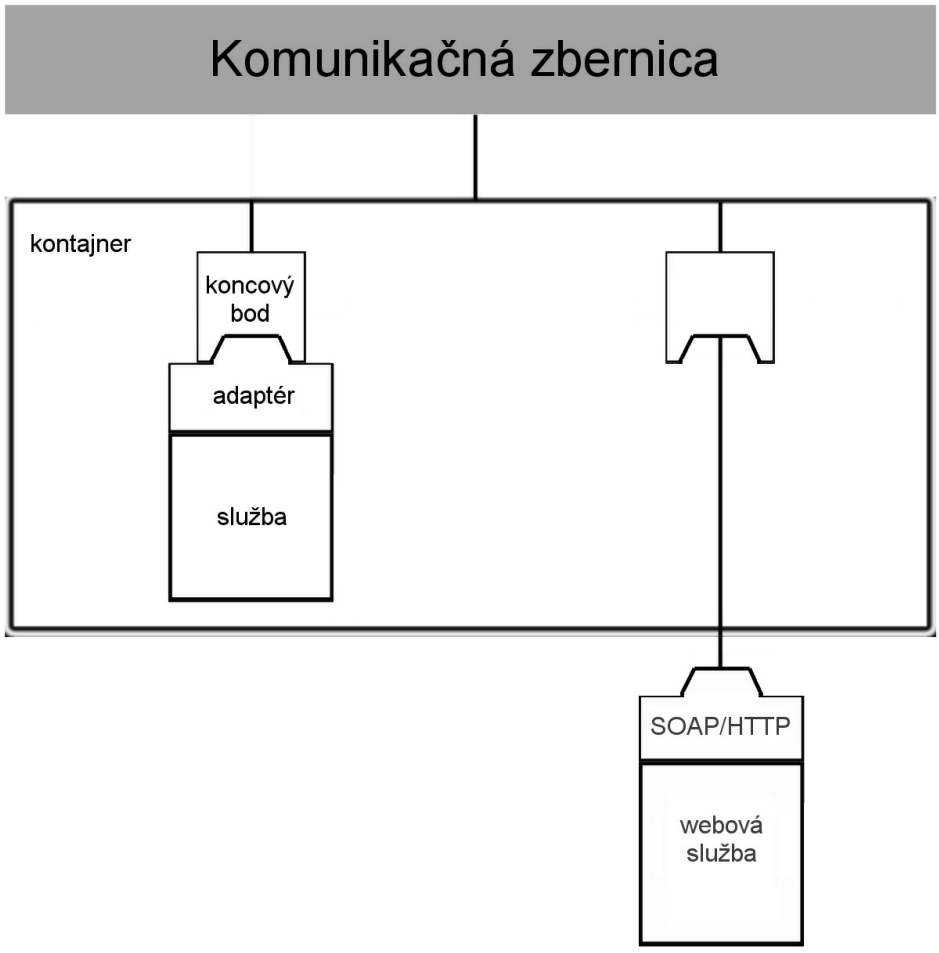
3.4 Architektúra ESB

V tejto kapitole si priblížime základné stavebné prvky architektúry ESB a pozrieme sa na ESB ako na trojúrovňovú architektúru, kde každá úroveň alebo vrstva využíva funkcionalitu nižšej vrstvy a poskytuje funkcionalitu pre vyššiu vrstvu.

Základnými stavebnými prvkami architektúry ESB sú spoločná komunikačná zbernica, služby a službové kontajnery. Keďže sa architektúra ESB radí medzi architektúry realizujúce koncept SOA, funkcionalita aplikácií je zverejnená ako jedna alebo viacero služieb a stačí teda uvažovať o službách.[8]

Jednotlivé služby sa pripájajú na komunikačnú zbernicu a prostredníctvom nej komunikujú posielaním správ. Služba je vždy nasadená v nejakom službovom kontajneri, ktorý spravuje jej životný cyklus a hlavne poskytuje službe rozhranie pomocou ktorého sa pripája na komunikačnú zbernicu, toto rozhranie sa volá abstraktný koncový bod. Rozhranie abstraktného koncového bodu môže reprezentovať priame napojenie na aplikačný adaptér alebo externé volanie, napríklad webovej služby (viď Obrázok 5 - Architektúra ESB). Komunikačnej zbernici sa hlbšie venujeme v kapitole 3.4.1 Úroveň správ. Službám, službovým kontajnerom a abstraktným koncovým bodom sa hlbšie sa venujeme v kapitole 3.4.2 Úroveň služieb.

[8] V texte práce na niektorých miestach používame pojmy aplikácia alebo systém s významom pojmu služba. Tam, kde treba odlíšiť aplikáciu, či systém od služby tak explicitne uvádzame.



Obrázok 5 - Architektúra ESB

Službový kontajner, niekedy tiež označovaný ako kontajner ESB, je koncepčne súčasťou zbernice. Na ESB sa dá pozerať aj ako na sieť prepojených službových kontajnerov, kde každý kontajner poskytuje funkcionalitu službám, ktoré sú v ňom nasadené. Službové kontajnery teda tvoria subsystémy zbernice v zmysle definície b) z kapitoly 3.2 Hľadanie definície ESB.

Úlohou integračného architekta pracujúceho s ESB je prepájanie existujúcich služieb a aplikácií pomocou zbernice tak, aby plnili nejakú úlohu definovanú napríklad procesným tokom. Na ESB sa teda dá pozerať na trochu úrovniach: úroveň správ, úroveň služieb a úroveň procesov.

3.4.1 Úroveň správ

Formát a štruktúra správ

Správy v ESB sú často vo formáte XML, ktorý poskytuje rad výhod, ktoré sme spomínali už v kapitole 3.2.3 Podniková zbernica služieb (ESB), napr. jeho nezávislosť na prostredí a širokú podporu v aplikáciách a programovacích jazykoch. Okrem XML však môžu byť správy v produktoch ESB vo forme textu, serializovateľných binárnych objektoch alebo ľubovoľnom inom formáte. Najväčšou výhodou vlastného formátu oproti XML je fakt, že práca s XML je výpočtovo zložitá.

Komunikačná zbernica

Jadrom ESB je komunikačná zbernica zaoberajúca sa správami. Od komunikačnej zbernice je požadované spoľahlivé a bezpečné asynchrónne posielanie správ pri definovateľnej kvalite služieb (*Quality of Sevice - QoS*) – napríklad môže byť požadované, aby každá správa dorazila do cieľa práve raz. Toto je zabezpečené podporou komunikácie *v* módoch *publish-subscribe* a bod-bod, škálovateľnosťou, schopnosťou ukladania odchádzajúcich správ v prípade nedostupnosti cieľovej služby (*store and forward)* a podobne. Bližší popis spomínaných módov sa nachádza napríklad v [PÁL06].

Takýmto jadrom môže byť napríklad niektorý z existujúcich produktov MOM, založených napríklad na špecifikácii JMS, WS-Reliability alebo WS-ReliableMessaging. Príkladom ESB produktov využívajúcich vrstvu MOM založenú na špecifikácii JMS môžu byť tri zo štyroch nami testovaných produktov: Sonic ESB, Open ESB aj Apache ServiceMix. Napríklad posledne menovaný využíva produkt Apache ActiveMQ [SMIX].

3.4.2 Úroveň služieb

Služba

Ako už bolo spomínané, na základe konceptov SOA je funkcionalita existujúcich aplikácií zverejnená ako jedna alebo viacero služieb. Tieto služby sú potom pripojené na komunikačnú zbernicu aby mohli spolupracovať s inými službami. Mnohé aplikácie sa však nemôžu prispôsobiť zbernici, preto musí ESB poskytovať adaptéry a konektivitu pre široké spektrum protokolov, pomocou ktorých sa aplikácie (služby) môžu na zbernicu pripojiť. Napríklad aplikácie umiestnené u obchodného partnera môžu vedieť komunikovať len prostredníctvom SOAP cez HTTP, zastarané (*legacy)* aplikácie môžu podporovať prepojenie s inými aplikáciami len pomocou prenosu súborov (txt, csv, vlastný formát) cez FTP, priamym prístupom cez TCP sockety, či vlastným rozhraním napísaným napríklad v jazyku C. K tomu všetkému treba pripočítať moderné spôsoby konektivity ako napríklad JMS, WS-Rel*, adaptéry na JDBC, SAP a mnoho iných produktov a technológií. Vidno teda, že pri porovnávaní ESB produktov môže byť kritériom aj počet a kvalita aplikačných adaptérov a druhov podporovanej konektivity.

Koncový bod

Abstrakciu nad konkrétnym protokolom, ktorým sa aplikácia (služba) pripája na zbernicu a miesto, na ktorom je služba umiestnená poskytuje práve abstraktný koncový bod. Abstraktný koncový bod je termín, ktorým sa súhrnne označujú všetky koncové body služby: vstupný koncový bod, výstupný koncový bod a prípadne ďalšie koncové body, ako chybový koncový bod, koncový bod pre odmietnuté správy alebo monitorovací koncový bod.

Služby sa navzájom adresujú pomocou abstraktných koncových bodov. Služba prijme správu zo svojho vstupného koncového bodu, vykoná svoju logiku a umiestni jednu alebo viacero správ na výstupný koncový bod. Môže sa pritom jednať o tú istú správu, modifikáciu pôvodnej správy, úplne novú správu, alebo aj viac správ – v závislosti na tom, akú funkcionalitu má služba vykonávať. Napríklad služba, ktorá vykonáva smerovanie na základe obsahu správy (*Content Based Routing - CBR*), zmení na základe obsahu správy cieľovú adresu (abstraktný koncový bod inej služby) v hlavičke správy a takúto výslednú správu umiestni na svoj výstupný koncový bod. Smerovanie na základe obsahu správy sa dá pri použití správ XML vyriešiť pomocou jednoduchej transformácie XSLT, ktorá zmení v hlavičke správy cieľovú adresu.

Iným príkladom môže byť služba, ktorá zaobstaráva objednávanie tovaru. Na vstupný koncový bod dostane správu so zoznamom objednávaného tovaru, tento rozdelí na jednotlivé položky, pre každú položku vytvorí novú správu a tú umiestni na výstupný koncový bod. O ďalší osud správ sa stará služobný kontajner, ktorý ich nasmeruje na zodpovedajúce cieľové adresy, kde môže ďalej nasledovať napríklad kontrola objednávaných položiek v skladových zásobách a podobne.

Služobný kontajner

Služobný kontajner je takpovediac základný kameň ESB. Ako už bolo povedané, služobný kontajner je koncepčne súčasťou zbernice. Jednotlivé služobné kontajnery medzi sebou komunikujú prostredníctvom správovej vrstvy (MOM). Služobný kontajner poskytuje pre služby, ktoré sú v ňom nasadené rozhranie koncových bodov, správu ich životného cyklu (naštartovanie (*startup*), ukončenie, vyčistenie použitých prostriedkov), nasadzovanie a podporu počas behu, združovanie vlákien (*thread pooling)*, spravovanie pripojení (*connection management*), ochranu a bezpečnosť, zabezpečuje požadovanú kvalitu služieb (QoS), transakčné služby a mnohé ďalšie. Niektoré z týchto služieb môže kontajner prenechať vrstve MOM, napríklad nastavenia QoS ohľadom doručovania správ (napríklad „najviac raz“). Vďaka kontajneru sa logika služieb nemusí týmito záležitosťami zaoberať a môže sústrediť na svoju funkcionalitu. Služobný kontajner navyše poskytuje monitorovaciu a spravovaciu funkcionalitu.

Ďalšou nespornou výhodou služobného kontajnera, ktorú vyzdvihuje [CHA04] a ktorý je v porovnaní s EAI hubmi alebo aplikačnými servermi dosť odľahčený (*lightweight)* je selektívne nasadzovanie funkcionality. Napríklad keď je v kontajneri na jednom uzle zbernice nasadená len jedna služba, povedzme smerovanie správ na základe obsahu, je veľkosť kontajnera relatívne malá a preto menej náročná na zdroje, zatiaľ čo pri „tradičných“ riešeniach by bolo potrebné na uzol nainštalovať celý zásobník sprostredkovateľskej funkcionality (*EAI broker stack*).

V Java Community Process bola schválená špecifikácia Java Business Integration (JBI), ktorá predstavuje štandard na prepájanie komponentov od rôznych výrobcov. Medzi komponenty sa radia služby aj kontajnery, JBI teda sľubuje možnosť štandardizovanej spolupráce kontajnerov medzi rôznymi implementáciami ESB. Viac o JBI v kapitole 3.7 Java Business Integration.

3.4.3 Úroveň procesov

Vyvolávanie jednotlivých služieb dostupných na zbernici môže prebiehať spôsobom podobným ako pri klient-server architektúrach, tj. trojicou operácií nájdi cieľovú službu, naviaž spojenie s cieľovou službou a vyvolaj operáciu na cieľovej službe („*find, bind, and invoke*“ [CHA04]). Pri takomto spôsobe vyvolávania však musí konzumentská služba poznať detaily o producentskej službe, obsahovať adresovaciu logiku a podobne.

Výhodnejšie je, keď konzumentské služby nevyvoláva priamo producentskú službu, ale toto volanie je súčasťou väčšieho udalosťami riadeného procesného toku. Tak sa docieli oddelenie aplikačnej logiky služby od vyvolávacej logiky. Operácie *find/bind/invoke* sa má na starosti službový kontajner a služba samotná sa môže sústrediť na prijatie správy zo vstupného koncového bodu, vykonanie svojej logiky a umiestnenie nejakej (potenciálne rovnakej) správy do výstupného koncového bodu. Ďalšie smerovanie správy zaobstará kontajner – môže správu poslať službe, ktorá pôvodne vyslala požiadavku (tzv. vzor *request-reply*) alebo inej službe na základe obsahu správy (tzv. vzor *reply-forward*). *Find/bind/invoke* správanie kontajnera sa nedefinuje písaním kódu, ale konfigurovaním. Môže sa jednať o choreografiu alebo sofistikovaný orchestrovaný procesný tok. Rozdiel medzi týmito dvomi druhmi procesných tokov je niekedy veľmi jemný, viaceré zdroje dokonca uvažujú len o jednom druhu procesného toku a pojmy choreografia a orchestrácia používajú ako synonymá.

Choreografia

Choreografia sa tiež nazýva smerovanie podľa itinerára a je riadené kontajnerom, resp. kontajnermi. Pod itinerárom rozumieme postupnosť diskrétnych smerovacích operácií. Napríklad pri spracovávaní objednávky môže itinerár pozostávať zo štyroch operácií: skontrolovanie kreditu, skontrolovanie skladových zásob, vybavenie objednávky, vystavenie faktúry. Detaily itinerára môžu byť uložené ako XML metadáta a posielané spolu so správou. Tým, že sa so správou prenáša aj jej itinerár, odpadá nutnosť centralizovaného riadenia a riadenie je distribuované medzi jednotlivé kontajnery smerujúce správy. Dôležité je, že služby zapojené do choreografie o tom nevedia, teda nemusia byť nijakým spôsobom prispôsobené choreografii.

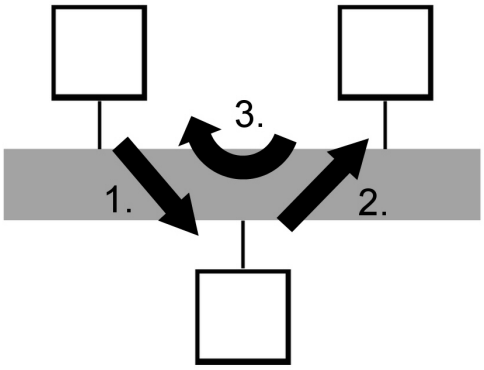
Na obrázku Obrázok 6 - Choreografia je znázornených niekoľko služieb pripojených na zbernicu, ktoré komunikujú na základe choreografie. Ako vidno z obrázku, komunikácia nie je riadená centrálne.

Orchestrácia

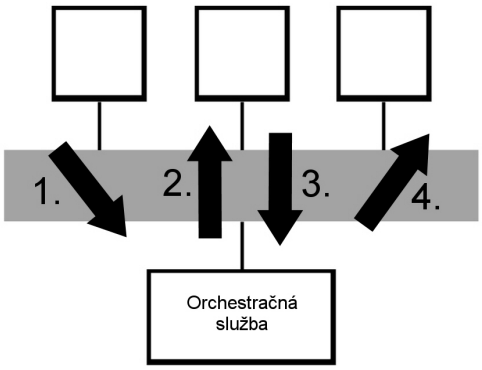
V mnohých prípadoch je však potrebné sofistikovanejšie riadenie ako itinerár, takéto riadenie sa nazýva orchestrácia a je čiastočne centralizované. Je riadené orchestračnou

službou pripojenou na zbernicu, ktorá má nadefinované procesné pravidlá a na základe týchto pravidiel riadi tok správ medzi niekoľkými službami. Orchestračná služba vie spravovať stavové informácie o procese v dlhšom čase, ako by to bolo možné pri itinerári. Je možné nadefinovať pravidlá pre zásah externých udalostí do toku akými sú napríklad ľudské akcie. Príkladom môže byť schválenie objednávky zodpovednou osobou pri prekročení nadefinovanej ceny. Ďalšou z úloh orchestračnej služby môže byť korelácia požiadaviek a odpovedí pri viacerých dlhších asynchrónnych konverzáciách - úloha ktorá by sa pomocou itinerára a správových front robila obtiažne. Orchestračné pravidlá môžu byť definované v nejakom procesnom modelovacom jazyku, vhodným štandardom môže byť napríklad BPEL, ktorému sa viac venujeme v kapitole 4.3.3 Úroveň procesov pri produkte Open ESB.

Na obrázku Obrázok 7 - Orchestrácia je znázornených niekoľko služieb pripojených na zbernicu, komunikácia medzi týmito službami je riadená centrálne pomocou orchestračnej služby.



Obrázok 6 - Choreografia



Obrázok 7 - Orchestrácia

3.5 Porovnanie ESB s inými integračnými architektúrami

Ako sme už spomínali v kapitolách 3.2 Hľadanie definície ESB a 3.3 Vývoj ESB, architektúra ESB vznikla zo skúseností získaných pri predchádzajúcich integračných architektúrach. Každá z predchádzajúcich architektúr mala svoje silnejšie a slabšie stránky určené svojimi vlastnosťami, ako napríklad či je aplikačná logika oddelená od integračnej alebo naopak s ňou prepojená, či ide o centralizovanú alebo distribuovanú architektúru, či je komunikácia synchrónna alebo asynchrónna a pod.

Pozrime sa teraz bližšie na hlavné architektúry a techniky používané pri integrácii aplikácií.

3.5.1 RPC - Remote Procedure Call

Vzdialené volanie procedúr (Remote Procedure Call - RPC) umožňuje volanie procedúr, funkcií alebo metód objektov (spolu označovaných ako operácie), nachádzajúcich sa na vzdialených počítačoch veľmi podobným alebo rovnakým spôsobom ako tomu je pri lokálnych volaniach. Medzi technológie RPC sa radia

napríklad CORBA, Java RMI (Remote Method Invocation), DCOM, ActiveX, Sun-RPC, XML-RPC a jeho nasledovníci SOAP 1.0 a 1.1.

Komunikácia typu RPC je zväčša synchrónna a štruktúra programov využívajúcich RPC je sekvenčná, podobne ako pri programoch využívajúcich „bežné“ lokálne volania. Pri volaní teda program môže využiť okamžitú odpoveď, ktorá hovorí, či operácia prebehla úspešne. V distribuovanom systéme však musí program počítať s ďalšími možnými chybami oproti lokálnym volaniam, napríklad čo má program robiť, ak mu nedôjde žiadna odpoveď od vzdialeného systému. Pri operáciách využívajúcich viacero RPC volaní závisí úspešnosť celej operácie na úspešnosti každého jedného volania, tj. napríklad na dostupnosti každého subsystému v čase volania operácie. Toto môže jednak byť problém zabezpečiť so zväčšujúcou sa veľkosťou distribuovaného systému a navyše to zvyšuje nároky na nutné ošetrovanie chýb a spracovanie výnimočných stavov.

Podľa [CHA04] je ďalšou nevýhodou integrácie pomocou RPC pevné naviazanie aplikácií na seba (*tight coupling*), keďže ľubovoľné dve komunikujúce aplikácie potrebujú poznať veľmi detailné informácie o svojich rozhraniach. So vzrastajúcim počtom aplikácií narastá aj počet rozhraní, ktoré treba udržovať. V najhoršom prípade dostávame analógiu s počtom hrán v kompletnom n-vrcholovom grafe, kde n je počet rozhraní. Treba však poznamenať, že problém s pevnou väzbou môže nastať aj pri nevhodnom návrhu integrácie založenej na správach namiesto operácií.

Ukazuje sa teda, že výhodnejšie býva integráciu založiť na voľne prepojených rozhraniach, napríklad pomocou vrstvy jedného alebo viacerých centrálnych serverov (ako tomu je pri *hub-and-spoke* architektúrach), či zbernice (ako tomu je pri ESB). Navyše, distribuovaný systém je výhodnejšie postaviť nad udalosťami riadeným asynchrónnym modelom.

3.5.2 MOM - Message Oriented Middleware

Idea komunikačnej zbernice pri ESB sa podobá na architektúry typu MOM (Message Oriented Middleware), založené napríklad na špecifikácii JMS (Java Message Service). Koncept architektúr typu MOM zahŕňa posielanie dát medzi aplikáciami vo forme správ, zväčša asynchrónnym spôsobom. Vďaka tomu možno aplikácie abstraktne oddeliť – posielajúca a prijímajúca aplikácia sa nemusia poznať a ani si nemusia byť vedomé tej druhej. Namiesto toho prijímajú a posielajú správy z a do komunikačného systému, ktorý dané správy doručí druhej strane. Aplikácie komunikujú s komunikačným systémom pomocou API určeného výrobcom konkrétneho produktu typu MOM, alebo pomocou nejakého štandardného API, napríklad JMS. Komunikačný systém je zväčša implementovaný ako správový server alebo správový sprostredkovateľ (*message broker)*.

ESB však nad takouto komunikačnou vrstvou ponúka navyše ešte vrstvu služieb, prípadne vrstvu procesov, čo uvádza ako hlavnú výhodu aj [CHA04]. Zatiaľ čo MOM umožňuje voľne prepájať aplikácie asynchrónnym spôsobom, samotné prepojenie je „pevno“ naprogramované a mieša sa tak aplikačná logika s integračnou. Toto sa dá čiastočne odstrániť použitím vhodných programátorských techník, stále však chýba akási abstrakcia na úrovni prepájacej logiky. Navyše, nie všetky produkty typu MOM

sú skutočne distribuované v tom zmysle, že nie sú dostatočne schopné siahať za hranice jednej fyzickej siete.

Komunikačnej zbernici v ESB sa viac venujeme v kapitole 3.4.1 Úroveň správ.

3.5.3 EAI hub-and-spoke

Integračné riešenia typu EAI *hub-and-spoke* sa začali objavovať v polovici 90. rokov a sú postavené nad vrstvou MOM alebo nad aplikačným serverom. Dodávateľmi takýchto riešení sú spoločnosti ako SeeBeyond, webMethods, Vitria, IBM, TIBCO alebo BEA.

[CHA04] radí medzi výhody tradičných architektúr typu EAI *hub-and-spoke* oddelenie procesov od integračného kódu a jednoduchú centralizovanú správu napr. procesných pravidiel alebo smerovacích pravidiel. Táto centralizovanosť však je zároveň podľa [CHA04] alebo [BAK05] aj nevýhodou, keďže EAI hub je potenciálny bod zlyhania (*single point of failure*) celej architektúry. Pri federovaných *hub-and-spoke* architektúrach, ktoré nahrádzajú jeden centralizovaný server sieťou serverov zdieľajúcich spoločné pravidlá táto nevýhoda ustupuje, avšak na každom serveri musí byť nainštalovaná celá alebo aspoň výrazná časť integračného sprostredkovateľa. [CHA04] v tomto vidí hlavný rozdiel medzi ESB a architektúrou EAI *hub-and-spoke*: ESB vďaka svojej distribuovanej architektúre kontajnerov služieb ponúka možnosť selektívneho nasadzovania (*selective deployment*) len tej funkcionality integračného sprostredkovateľa, ktorá je potrebná na konkrétnom uzle (napríklad iba smerovanie na základe obsahu alebo iba XSLT transformácia).

Navyše sa ukázalo, že riešenia založené typu *hub-and-spoke* majú problém so škálovateľnosťou mimo fyzické LAN segmenty. Zvyčajne ide o monolitické, drahšie systémy od jedného dodávateľa založené na vlastných štandardoch. Naproti tomu ESB je navrhnuté ako distribuovaná integračná architektúra založená na otvorených štandardoch, ktorej cieľom je poskytnúť lacnejšie riešenia. Niektoré riešenia typu ESB, hlavne komerčné, sú však stále závislé na svojom dodávateľovi a nie sú plne prepojiteľné s ľubovoľnými komponentmi od iných dodávateľov. Návrh štandardu, ktorý sa snaží riešiť tento problém je Java Business Integration, spomínaný v kapitole 3.7 Java Business Integration.

3.5.4 Webové služby

Webové služby (*web services*) ukázali zmysluplnosť SOA v tom, že poskytli spôsob spolupráce medzi aplikáciami založený na štandardoch. Vďaka abstrakcii na úrovni služieb je možné prepájať aplikácie na rôznych platformách a prostrediach.

Webové služby sa dajú použiť rôznymi spôsobmi. Jednou z možností je použitie webových služieb ako nástroja pre RPC, základnou komunikačnou jednotkou je v tomto prípade operácia WSDL. Táto možnosť použitia je však kritizovaná za to, že vedie k pevným prepojeniam, ktorých nevýhodám sme sa už venovali v kapitole 3.5.1 RPC - Remote Procedure Call.

Druhou a výhodnejšou možnosťou je použitie webových služieb na implementáciu princípov SOA, základnou komunikačnou jednotkou je v takomto prípade správa, nie

operácia, čo podporuje voľné prepojenie koncových aplikácií. K historicky starším špecifikáciám pre „jadro“ webových služieb patria špecifikácie pre definovanie služieb pomocou WSDL, definovanie štruktúry správ XML pomocou SOAP, adresárové služby pomocou UDDI. K týmto neskôr začali pribúdať ďalšie špecifikácie riešiace otázky ako bezpečnosť alebo spoľahlivosť komunikácie. Medzi tieto „novšie“ špecifikácie patria: WS-Security (bezpečnosť), WS-Reliability, WS-ReliableMessaging (spoľahlivosť), WS-Addessing (adresovanie služieb pomocou koncových bodov), WS-Notification (udalosťami riadené prepájanie webových služieb) a mnohé ďalšie, súborne označované ako WS-*. Štandardom WS-* sa venuje napríklad diplomová práca [BIS06].

ESB vychádza z princípov webových služieb, podporuje zrelé a overené štandardy ako XML, SOAP, či WSDL a taktiež by malo byť schopné podporovať novovznikajúce štandardy WS-*, aby jednak umožňovalo integráciu s aplikáciami používajúcimi tieto štandardy a jednak samo vedelo vyťažiť z možností, ktoré tieto štandardy poskytujú (bezpečnosť, adresovanie, asynchrónna komunikácia, atď.). ESB teda poskytuje podporu aj pre integráciu aplikácií založených na webových službách, navyše ponúka možnosti a nástroje na správu a konfiguráciu, procesné riadenie a podobne. ESB sa teda dá chápať ako vrstva zastrešujúca okrem iných aj technológií aj webové služby ([CAPTECH], [SONDOC]).

3.6 Výhody a nevýhody ESB

Ako každá architektúra, aj ESB má svoje výhody a nevýhody:

Medzi výhody ESB patrí jej distribuovaná kontajnerová architektúra, ktorá zaručuje ľahké postupné nasadzovanie a integráciu aplikácií a škálovateľnosť zbernice. ESB sa neobmedzuje na veľké alebo malé integračné projekty. Pri vhodnej granularite služieb podporuje voľné prepojenia aplikácií. Je kladený dôraz na konfigurovateľnosť namiesto písania kódu, čo zaručuje kratšiu dobu vývoja integračného projektu a lepšiu opakovanú použiteľnosť. ESB je založené na zaužívaných štandardoch (napr. XML, SOAP, WSDL, JMS), podporuje nové štandardy (napr. JBI, BPEL) a umožňuje rozšíriteľnosť o novo vzniknuté štandardy (napr. WS-*). Pri použití transparentných formátov ako XML je umožnené ľahké monitorovanie a audit procesov. Na úrovni zbernice je riešená bezpečnosť, kvalita služieb a spoľahlivosť. Podporuje rôzne spôsoby procesného riadenia – od itinerárového až po sofistikovaný procesný tok, pričom komponent riadiaci procesný tok sa vďaka zapojiteľnej architektúre (*plugin architecture)* pripája štandardným spôsobom na zbernicu, rovnako ako iné komponenty. Vďaka konfigurovateľnosti môže výsledné integračné riešenie využívať metodiky integračných vzorov, napríklad vzoru VETRO – *Validate, Enrich, Transform, Route, Operate*. V neposlednom rade je to cena – sú dostupné kvalitné open source produkty a aj cena komerčných produktov je napríklad oproti tradičným EAI hubom nižšia.

Medzi nevýhody ESB patrí čiastočná závislosť na konkrétnej implementácii, keďže architektúra ESB nie je daná nejakou presnou špecifikáciou. Pri menších integračných projektoch môže byť vrstva služieb príliš komplikované riešenie, ak by postačovala vhodne navrhnutá integrácia pomocou správ. Naopak, pri nevhodne navrhnutej štruktúre správ aj pri použití ESB môže dôjsť k pevným väzbám medzi aplikáciami a problémom s udržovaním rôznych verzií jednej správy. Správy vo formáte XML

môžu predstavovať nevýhodu v rýchlosti spracovania. Pri použití orchestrovaných procesných tokov vyžaduje mať v organizácii dobre definované procesy. Konfigurácia ESB vyžaduje od architekta nové schopnosti, niektoré produkty sú vďaka svojej rozsiahlej funkčnosti časovo náročné na zvládnutie. Nie všetky open source produkty majú dostatočnú dokumentáciu (čo je ale všeobecný problém open source produktov). Z týchto všetkých dôvodov môže byť návratnosť investícií vložených do ESB posunutá do neskorších projektov, keď sa získajú dostatočné skúsenosti.

3.7 Java Business Integration

Java Business Integration je jedna z mnohých Java technológií, ktorých princípy sú použiteľné pri vytváraní ESB, prípadne sa s ESB princípmi priamo prelínajú. Ďalšími Java technológiami môžu byť napríklad J2EE Connector Architecture na prepájanie existujúcich, často zastaraných aplikácií alebo Java Management eXtensions nástroje na štandardizovanú správu a monitorovanie ľubovoľnej aplikácie podporujúcej JMX.

Špecifikácia Java Business Integration (ďalej len JBI), označovaná aj ako JSR 208 sa zaoberá zapojiteľnou (*pluggable)* komponentovou architektúrou, pomocou ktorej môžu štandardným spôsobom spolupracovať komponenty od rôznych dodávateľov. Architektúru JBI spomína aj David Chappell vo svojej prezentácii [CHA05JBI]. David Chappel je autor [CHA04], ktorá bola v čase písania tejto práce jediná tlačená publikácia o ESB. V [CHA05JBI] hovorí o JBI:

> „JBI pomôže formovať spôsob integrácie a ESB podobne ako to robila špecifikácia EJB pre aplikačnú logiku a aplikačné servery v druhej polovici 90. rokov.“

Špecifikácia vznikla v marci 2003, v júni 2005 bola schválené pomocou Java Community Process a od augusta 2005 je jej finálna verzia 1.0 dostupná na [JSRJBI]. Do JBI Expert Group, ktorá návrh špecifikácie vytvorila patria spoločnosti ako BEA, Borland, IBM, Nokia, Novell, Oracle, SAP, SeeBeyond, Sonic Software, Sun Microsystems, Sybase, TIBCO, WebMethods a mnohé ďalšie.

Špecifikáciu JBI priamo implementujú aj dva open source produkty ESB, ktorým sa v tejto práci venujeme: Open ESB aj Apache ServiceMix. Ďalšie dva produkty (Sonic ESB a Mule) podporujú integráciu s prostrediami založenými na JBI, napríklad podporujú nasadenie svojich služieb do kontajnerov JBI. Dobrý prehľad architektúry JBI sa nachádza napríklad v dokumentácii k Apache ServiceMix [SMIXJBI].

V záujme konzistentnosti s popisom architektúry ESB sa na architektúru JBI môžme pozerať tiež na troch úrovniach: úroveň správ, úroveň služieb a úroveň procesov. Neskôr sa o tieto úrovne budeme opierať pri popise produktov OpenESB (kapitola 4.3 Open ESB) a Apache ServiceMix (4.4 Apache ServiceMix).

3.7.1 Úroveň správ

Formát a štruktúra správy

Komunikácia medzi komponentmi prebieha prostredníctvom normalizovaných XML správ. Každá správa sa skladá z hlavičky (metadát) a tela. V hlavičke sa nachádzajú dáta asociované so správou, zaoberajúce sa napr. bezpečnosťou alebo transakciami. Telo správy je XML dokument, ktorého štruktúra je definovaná napr. schémou XML. Správa môže obsahovať aj prílohy, ktoré nie sú XML dokumenty a s ktorými sa pracuje pomocou spracovávačov (*handlers*) definovaných v správe. Vďaka použitiu jednotného normalizovaného formátu správ sa zjednodušuje napríklad proces smerovania.

Komunikácia medzi komponentmi môže prebiehať v štyroch módoch prevzatých z WSDL 2.0 (*MEP – Message Exchange Patterns*):

- **Jednosmerná** (*in-only)*: konzument pošle správu, producent vráti len stavovú odpoveď (napríklad „ukončenie komunikácie“).

- **Spoľahlivá jednosmerná** (*robust in-only)*: Spoľahlivé jednosmerné posielanie. Konzument pošle správu, producent vráti stavovú odpoveď alebo chybu. Ak je odpoveď chybová, konzument tiež pošle stavovú odpoveď.

- **Požiadavka - odpoveď** (*in-out)*: Konzument pošle správu, producent odpovedá správou alebo chybou, konzument odpovedá stavovou odpoveďou.

- **Požiadavka - nepovinná odpoveď** (*in optional-out*): Obojsmerná, kde odpoveď producenta je nepovinná.

Od implementácie služieb sa očakáva, že budú podporovať komunikáciu v týchto módoch, na základe čoho sa dajú štandardným spôsobom prepojiť.

Komunikačná zbernica

Špecifikácia JBI sa venuje hlavne architektúre kontajnera (ktorý popisujeme v ďalšej kapitole), ideu napojenia kontajnerov na komunikačnú zbernicu v ESB spomína len okrajovo. Ako uvedieme v nasledujúcej kapitole, o funkcionalitu prepájania jednotlivých kontajnerov sa starajú špeciálne komponenty v rámci kontajnera, nazývané prepájacie komponenty (*binding components - BC*).

Istý druhom komunikačnej zbernice, avšak pracujúcej vnútri kontajnera je smerovač normalizovaných správ (N*ormalized Message Router - NMR*), ktorému sa hlbšie venujeme tiež v nasledovnej kapitole.

3.7.2 Úroveň služieb

Na úrovni služieb sa pozrieme bližšie na služby (komponenty), koncové body, pomocou ktorých komunikujú a samotný službový kontajner. V závere popíšeme možnosť spravovania komponentov a kontajnerov JBI.

Služba

Producenti a konzumenti služieb sú v JBI súhrnne označovaní ako komponenty, ponúkané služby sú popísané pomocou WSDL 1.1 alebo 2.0. Komponenty môžu byť dvoch druhov: službové prostriedky (*Service Engine - SE*) a prepájacie komponenty (*Binding Components - BC*). Službové prostriedky (SE) poskytujú a využívajú logiku a transformačné služby. SE pracujú ako kontajnery, do ktorých môžu byť nasadené služby definované pomocou WSDL. Prepájacie komponenty (BC) sa venujú transformácii normalizovaných správ na konkrétny komunikačný protokol (napr. JMS) a posielaniu a prijímaniu správ pomocou tohto protokolu. BC teda poskytujú pripojenie na služby mimo kontajneru JBI. Rozdelenie komponentov na dva druhy slúži na lepšie oddelenie aplikačnej logiky (SE) od komunikačnej (BC), čo znižuje zložitosť a zvyšuje flexibilitu riešenia.

Komponenty JBI môžu pochádzať od rôznych dodávateľov a vďaka črtám architektúry JBI dokážu spolu bez problémov komunikovať a spolupracovať. Mnoho komponentov je vyvíjaných s otvorených zdrojovým kódom, napríklad v rámci projektu Open JBI Components[9]. Komponenty z tohto projektu využíva produkt Open ESB, ktorému sa hlbšie venujeme v kapitole 4.3 Open ESB.

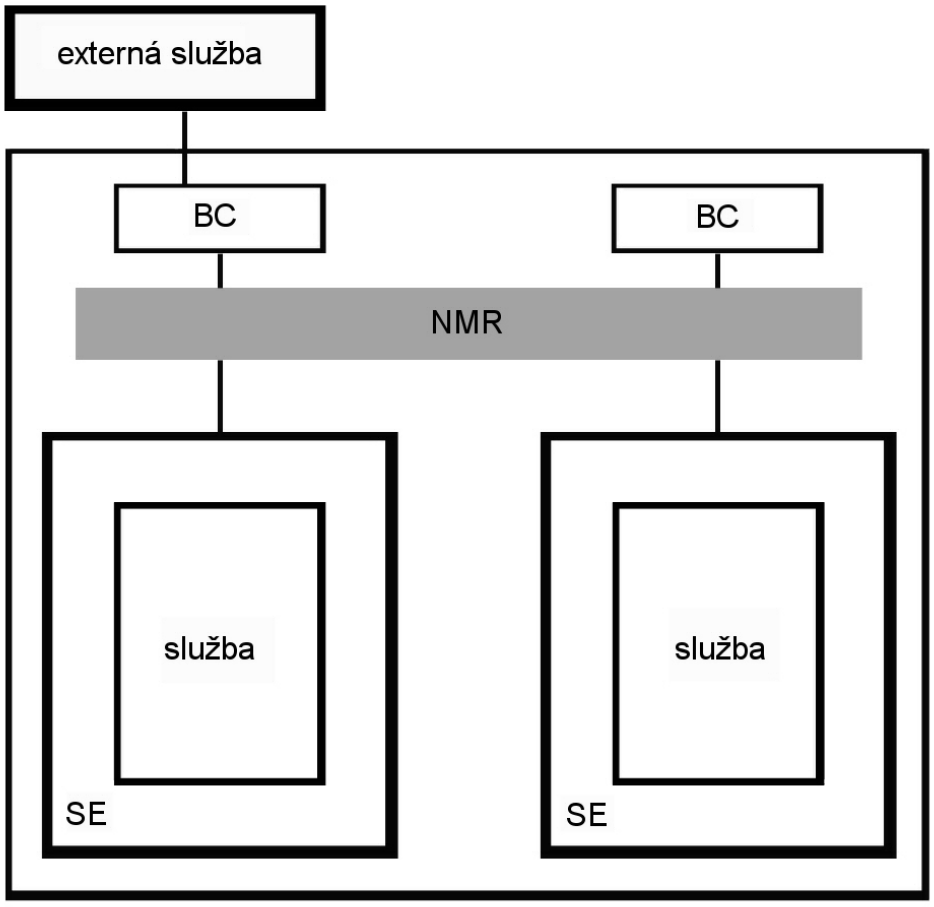
Koncový bod

JBI využíva koncept koncových bodov prevzatý z WSDL 2.0. Koncové body pomocou ktorých služby v JBI prijímajú a odosielajú správy môžu byť dvoch druhov – interné a externé. Pomocou interných koncových bodov sa posielajú správy prostredníctvom smerovača normalizovaných správ ostatným komponentom v kontajneri, externé koncové body slúžia na komunikáciu so službami mimo kontajneru JBI. Úlohou prepájacích komponentov (BC) je preklad medzi internými a externými koncovými bodmi.

Službový kontajner

Službový kontajner poskytuje pomocou rozhrania JMX funkcionalitu na nasadzovanie, spravovanie a monitorovanie komponentov. Spravuje životný cyklus komponentov, ktoré sú v ňom nasadené. Jednou z hlavných úloh službového kontajnera je však sprostredkovanie služieb smerovača normalizovaných správ (*Normalized Message Router*), ktorý poskytuje funkcionalitu pre smerovanie správ medzi komponentmi v rámci jedného kontajneru JBI. Služba môže inú službu adresovať pomocou jej koncového bodu alebo pomocou mena služby. V takom prípade smerovač normalizovaných správ zabezpečí nájdenie správnej služby a jej koncového bodu. Na obrázku Obrázok 8 - Architektúra JBI je znázornená architektúra kontajnera tak, ako je popísaná v špecifikácii.

[9] https://open-jbi-components.dev.java.net



Obrázok 8 - Architektúra JBI

Kontajner JBI sa dá chápať ako jedna z možných implementácií službového kontajnera ESB. [CHA04] alebo [GOP06] vidia možnosť použiť kontajner JBI ako kontajner ESB, ktorý sa pomocou prepájacích komponentov (BC) pripája na spoločnú komunikačnú zbernicu. Na podobnom princípe pracujú aj dva open source produkty implementujúce špecifikáciu JBI (Open ESB a Apache ServiceMix), ktoré analyzujeme v tejto práci. Každý produkt ponúka implementáciu kontajnera JBI, jednotlivé kontajnery používajú na komunikáciu nejaký produkt typu MOM.

Konfigurácia a správa

O životný cyklus komponentov sa stará kontajner JBI. Tento poskytuje podporu pre správu prostredia JBI pomocou štandardných administračných nástrojov JMX, ktoré umožňujú nasadzovanie (inštaláciu) komponentov, ovládanie životného cyklu komponentov, monitorovanie komponentov alebo nasadzovanie artefaktov do komponentov. Artefaktom môže byť napríklad konkrétny štýl XSLT (*XSLT styleheet*), ktorý sa môže nasadiť do komponentu SE zaoberajúceho sa transformáciami XSLT. Týmto začne komponent SE poskytovať novú funkcionalitu, je tu teda vidno analógiu s nasadzovaním komponentov do kontajnerov JBI. Aby sa

tieto dva procesy odlíšili, hovorí sa v návrhu špecifikácie JBI pri komponentoch o inštalácii, zatiaľ čo pri artefaktoch o nasadzovaní.

3.7.3 Úroveň procesov

Choreografia

Špecifikácia JBI sa explicitne nevenuje úrovni choreografie. Každá komunikácia prebieha na najzákladnejšej úrovni medzi dvoma službami (producentom a konzumentom) v niektorom zo štyroch módov MEP. Komunikáciu vždy začína konzument. O nájdenie vhodného producenta sa stará NMR. JBI sa však nevenuje komunikácii, pri ktorej by jeden konzument využíval viacero producentov, ku ktorým by boli správy smerované na základe itinerára, ktorý by v sebe obsahovali.

Orchestrácia

Podobne ako pri choreografii, špecifikácia JBI nespomína ani orchestrované procesy. O procesnú orchestráciu sa stará implementácia služobného prostriedku (SE) pre vhodný procesný jazyk, napríklad BPEL. Takýto služobný prostriedok pracuje ako kontajner, do ktorého sa nasadzujú jednotlivé procesy.

4 Analyzované produkty

V súčasnosti prebieha v produktovej kategórii ESB situácia, ktorá ťaží z nejasnej definície ESB. Viacero firiem si pripisuje zásluhy na tom, že ich produkt typu ESB bol na trhu prvý (Sonic, CapeClear), prípadne tvrdia, že konkurenčné produkty nie sú skutočné ESB (CapeClear) alebo že ich produkt poskytuje viac funkcionality ako akékoľvek ESB (Microsoft). Navyše sa na trhu objavujú aj open source produkty, ktoré sú už v mnohých ohľadoch porovnateľné s komerčnými. V prílohe A: Prehľad komerčných a open source produktov typu ESB, október 2006 prinášame zhrnutie hlavných komerčných a open source produktov typu ESB. Tabuľky sú prevzaté z [FRY06].

Keďže nie je v našich možnostiach spraviť vyčerpávajúce porovnanie jednotlivých produktov, zamerali sme sa na jeden komerčný (Sonic ESB) a tri open source produkty (OpenESB, Apache ServiceMix a Mule).

4.1 Kritériá porovnávania

K porovnávaniu jednotlivých ESB produktov môžeme pristupovať dvoma spôsobmi: "zdola" a „zhora". Pri porovnávaní zdola sledujeme podobnosti a odlišnosti medzi produktmi, napríklad akú vrstvu MOM používajú, aké sú podporované adaptéry a podobne. Do porovnávania zdola patria aj výkonnostné testy. Pri porovnávaní zhora sa sústredíme na to, čo od produktov potrebujeme, čiže akým spôsobom konkrétny produkt realizuje naše požiadavky identifikované v kapitole 2 Integrácia v informačnom systéme Univerzity Komenského. Nie je však cieľom porovnávania určiť, ktorý produkt je lepší a ktorý horší, skôr prehľadným spôsobom produkty porovnať a zhrnúť pozitíva a negatíva každého produktu.

Ku každému produktu uvedieme stručný všeobecný popis, popíšeme každú z troch úrovní architektúry ESB (úroveň správ, úroveň služieb a úroveň procesov) a zameriame sa na realizáciu požiadaviek pre integráciu v prostredí UK. Nakoniec zhrnieme výhody a nevýhody konkrétneho produktu.

Požiadavky identifikované v kapitole 2 Integrácia v informačnom systéme Univerzity Komenského zhrnieme do nasledovných piatich kategórií:

- **Rôznorodosť systémov a prepojení**

  V tejto kategórii nás zaujíma, aké adaptéry a spôsoby napojenia na iné systémy, komunikačné protokoly a štandardy daný produkt podporuje. Z prílohy B: Rozhrania systémov v rámci IS UK sa dá vyčítať, že najdôležitejšie sú adaptéry na prácu s databázou, lokálnym súborovým systémom a správami JMS. Zaujímajú nás ale aj ostatné adaptéry, ktoré môžu byť dôležité pri budúcom rozširovaní. Taktiež nás v tejto kategórii zaujíma aké preddefinované služby poskytuje a aké má obmedzenia na veľkosť správ.

- **Robustnosť, dostupnosť, odozva, súčasné prístupy**

  V tejto kategórii nás zaujíma akým spôsobom rieši daný produkt problém dostupnosti a robustnosti pomocou škálovateľnosti, tj. možnosti použitia

viacerých inštancií služieb, kontajnerov alebo serverov na vyvažovanie záťaže (*load balancing)* – či už záťaže na úrovni správ alebo na úrovni služieb.

- **Bezpečnosť, autentifikácia, autorizácia**

  V tejto kategórii nás zaujíma aké autorizačné a autentifikačné protokoly daný produkt podporuje (napr. SSL, WS-Security, WS-Rel*, Kerberos, protokoly pre Public Key Infrastructure – napr. X.509), či podporuje napojenie na LDAP server (Stručný popis jednotlivých protokolov sa nachádza v kapitole Slovník pojmov).

- **Správa, monitorovanie, logging, auditing, zálohovanie**

  V tejto kategórii nás zaujíma, aké nástroje na správu, zaznamenávanie (*logging*), monitorovanie a auditing daný produkt používa, či sú založené na štandardoch ako napr. JMX a pod.

  Na automatické zálohovanie dát treba implementovať službu, ktorá bude využívať možnosti konkrétnej databázy (*dump*, export a pod), preto zálohu dát pri produktoch do hĺbky neskúmame.

- **Modulárnosť**

  Požiadavke modulárnosti dobre vyhovuje zapojiteľná architektúra ESB a spôsob postupnej integrácie, ktorý ESB podporuje. Modulárnosť je vo všeobecnosti črtou architektúr realizujúcich princípy SOA, nakoľko vieme implementáciu služby nahradiť inou, bez nutnosti meniť ostatné služby, ktoré využívajú jej funkcionalitu. Vďaka architektúre ESB vieme ľahko pripájať nové služby a prepájať ich s existujúcimi pomocou konfigurácie služieb alebo kontajnerov.

  V tejto kategórii teda ide o všeobecné vlastnosti architektúry ESB, preto sa jej pri konkrétnych produktoch hlbšie nevenujeme.

- **Univerzálne číselníky**

  Podobne ako pri predchádzajúcej kategórii, ani univerzálne číselníky nie sú funkcionalita, ktorá by bola závislá na konkrétnom produkte, preto sa tejto kategórii pri konkrétnych produktoch hlbšie nevenujeme. V každom produkte by sa malo dať implementovať rozhranie služieb pre univerzálne číselníky, a ľubovoľná služba pripojená k zbernici by dokázala univerzálne číselníky využiť.

4.2 Sonic ESB

Sonic ESB je jeden z produktov rodiny Sonic ESB Product Family, ktorú ponúka spoločnosť Progress Software. Ide o komerčný produkt s históriou siahajúcou do roku 2002 [FRY06]. Ďalšie produkty Sonic ESB Product Family sú Sonic Workbench - vývojárske, konfiguračné a administratívne prostredie založené na platforme Eclipse, Sonic Orchestration Server - orchestračná služba zapojiteľná do spoločnej

infraštruktúry, Sonic XML Server - optimalizované spracovanie XML a ukladanie XML, Sonic Database Service - zjednodušený prístup k relačným databázam, Adapters for Sonic ESB - asi 300 adaptérov na rozličné systémy, Sonic Collaboration Server - integrácia s externými partnermi pomocou *business-to-business* (B2B) protokolov a webových služieb. Sonic ESB je založené na programovacom jazyku Java, na svoj beh vyžaduje JDK 1.4.2 alebo vyššie. Sonic ESB podporuje platformy MS Windows, Sun Solaris, Linux, IBM AIX a HP-UX.

V mesiacoch september a október 2006 sme testovali verziu Sonic ESB 7.0, Sonic Orchestration Server 7.0 a Sonic Workbench 7.0. V apríli 2007 vydal Progress Software verziu 7.5, ktorá prináša niekoľko rozšírení, napríklad Sonic BPEL Server, podporu pre skriptované nasadzovanie pomocou Ant a rôzne rozšírenia v oblasti bezpečnosti, či spoľahlivosti.

Hlavným zdrojom informácií v tejto kapitole je dokumentácia dostupná na internete ([SONDOC]) a interná dokumentácia z testovanej verzie produktov ([SONDEV]). Obidva pramene poskytujú pomerne rozsiahle zdroje informácií, často sa v nich však prelína celkový „manažérsky“ pohľad s hĺbkovým „vývojárskym“. Toto čiastočne obmedzuje čitateľnosť a pochopiteľnosť dokumentácie a s tým súvisiacu ťažkú naučiteľnosť tohto rozsiahleho produktu.

4.2.1 Úroveň správ

Formát a štruktúra správ

Sonic ESB podporuje správy v rôznych formátoch: v textovom formáte, vo formáte XML, v binárnom formáte a pod. Najvýhodnejšie je však používať formát XML, keďže existuje viacero služieb, ktoré pracujú s XML správami, napríklad transformácia pomocou XSLT alebo smerovanie na základe obsahu realizované cez výrazy písané v XQuery. Správy v Sonic ESB môžu obsahovať prílohy.

Služby v Sonic ESB pracujú vnútorne vždy s vlastným formátom označovaným ako XQMessage, konverzia z formátu zbernice do XQMessage a naopak sa deje na vstupnom a výstupnom koncovom bode služby.

Realizácia komunikačnej zbernice

Sonic ESB sa opiera o správovú vrstvu SonicMQ, ktorá sprostredkováva komunikáciu medzi jednotlivými službami. SonicMQ sa predáva aj ako samostatný produkt typu MOM, poskytujúci podporu napr. pre JMS, HTTP alebo SOAP. Architektúru SonicMQ tvorí jeden alebo viacero správových sprostredkovateľov, na najnižšej úrovni má teda Sonic ESB črty sprostredkovateľských architektúr. Tieto črty však nájdeme aj u ďalších dvoch open source produktoch (OpenESB a Apache ServiceMix), ktoré tiež využívajú existujúce produkty typu MOM na úrovni správ.

SonicMQ poskytuje funkcionalitu vyvažovania záťaže (*load balancing*), či *clustering* a taktiež zohráva podstatnú úlohu pri riešení bezpečnosti. Týmto funkcionalitám sa hlbšie venujeme nižšie v kapitole 4.2.4 Realizácia požiadaviek.

Pri komunikácii medzi službami v rámci jedného kontajnera sa môže správová vrstva SonicMQ obísť a komunikáciu zaobstaráva priamo služobný kontajner.

4.2.2 Úroveň služieb

Služba

Nad vrstvou SonicMQ je vybudovaná vrstva Sonic ESB, zaoberajúca sa službami (tiež označovanými ako „služby ESB“). Služby môžu pracovať samostatne alebo byť súčasťou nejakého procesu ESB, ktorý značí smerovanie podľa itinerára – viď nasledujúca kapitola. Služby sú nasadené v odľahčenom službovom kontajneri ESB, ktorý okrem iného spravuje ich životný cyklus. V jednom kontajneri môže byť nasadená jedna alebo viac služieb. Podobne ako pri JBI, na popis služieb sa používa jazyk WSDL, takýto popis je však nepovinný. Služby sú implementované sú v jazyku Java. Konfigurácie služieb sú uložené v adresárovej službe.

Preddefinované služby, ktoré sú sučasťou inštalácie Sonic ESB sú popísané v kapitole 4.2.4 Realizácia požiadaviek.

Koncový bod

Služby prijímajú a posielajú správy pomocou koncových bodov, o ktoré sa stará služobný kontajner, teda takpovediac prepája koncové body pomocou správovej zbernice SonicMQ. Koncové body sú štandardne realizované ako *JMS topics* v správovej zbernici SonicMQ.

Koncový bod služby v Sonic ESB sa zo všetkých analyzovaných produktov najviac podobá na abstraktný koncový bod z kapitoly 3.4.2 Úroveň služieb, nakoľko sa ďalej delí na vstupný koncový bod, výstupný koncový bod, chybový koncový bod a koncový bod pre zamietnuté správy.

Na poslanie správy stačí, ak služba umiestni do svojho výstupného koncového bodu (realizovaného ako schránka *outbox*) správu s cieľovou adresou, určujúcou iný koncový bod, inú službu v tom istom kontajneri (adresu kontajner interpretuje ako vstupný koncový bod služby) alebo proces ESB (adresu kontajner interpretuje ako vstupný bod do procesu).

Služobný kontajner

Služobný kontajner spravuje životný cyklus služieb, ktoré sú v ňom nasadené a poskytuje napojenie týchto služieb na komunikačnú zbernicu prostredníctvom koncových bodov, na spravovacie prostredie a adresárovú službu.

V prípade komunikácie medzi službami nasadenými v jednom kontajneri môže kontajner priamo smerovať správy medzi službami a obísť tak správovú zbernicu.

Konfigurácie a správa

Službový kontajner pripája služby pomocou protokolu JMX na spravovacie prostredie *Sonic management framework* a centrálnu adresárovú službu *Sonic Directory Service*, kde sú umiestnené konfiguračné dáta, XSLT *stylesheets*[10], pravidlá pre smerovanie na základe obsahu a podobne. Centrálna adresárová služba je pripojená na komunikačnú zbernicu prostredníctvom samostatnej inštancie správového sprostredkovateľa SonicMQ. Každý službový kontajner si ukladá používané konfigurácie získané z adresárovej služby do vyrovnávacej pamäte (*cache*). V prípade výpadku sprostredkovateľa adresárovej služby môže kontajner pracovať s dátami z vyrovnávacej pamäte, až pokiaľ nebude potrebovať doposiaľ nepoužitý zdroj, napríklad nejaký konkrétny XSLT *stylesheet* alebo môže byť použitá záložná kópia sprostredkovateľa s replikovanými dátami, tj. stratégia *failover*.

Konfigurácie služieb sú uložené v adresárovej službe vo formáte XML a pristupujú k nim jednotlivé inštancie služieb. Služby sa konfigurujú pomocou grafického nástroja, ktorý je súčasťou vývojárskeho prostredia Sonic Workbench.

4.2.3 Úroveň procesov

Choreografia

Procesy môžu byť realizované dvomi spôsobmi: ako procesy ESB alebo ako orchestračné procesy. Procesy ESB reprezentujú procesný tok smerovaný na základe itinerára (ide teda o choreografiu), sú podporované v produkte Sonic ESB. Itinerár môže pozostávať z krokov, ktoré vyvolávajú služby ESB, iné procesy ESB alebo externé webové služby.

Orchestrácia

Orchestračné procesy sú oproti procesom ESB sofistikovanejšie, stavové a s dlhou životnosťou. Na podporu pre orchestračné procesy je potrebné na zbernicu pripojiť orchestračnú službu, ktorá sa dodáva ako súčasť samostatného produktu Sonic Orchestration Server.

Sonic používa vlastný modelovací jazyk pre orchestračné procesy, nie je teda založený na otvorenom štandarde, akým je napríklad jazyk BPEL. Modelovací jazyk sa podobá na diagram aktivít v modelovacom jazyku UML, kde prechody medzi jednotlivými aktivitami sú riadené udalosťami, ako napríklad prijatie správy alebo vstup od používateľa. Aktivity reprezentujú proces ESB, službu ESB, adresu ESB alebo externú webovú službu. Orchestračné procesy sa modelujú pomocou grafického nástroja, ktorý je súčasťou vývojárskeho prostredia Sonic Workbench. V tomto nástroji sa dajú orchestračné procesy aj odladovať a spravovať.

V orchestračnej službe môže byť súčasne spustených viacero procesov a pre každý proces môže byť súčasne spustených viacero inštancií. Pre potreby škálovateľnosti (*scalability*) a vyvažovania zátaže (*load balancing*) môže byť orchestračná služba

[10] Viď kapitola Slovník pojmov.

nasadená viac krát. Súčasťou produktu Sonic Orchestration Server je okrem orchestračnej služby (*Orchestration Service*) aj služba na správu a meranie bežiacich procesov (*Process Search Service*) a služba na štandardný spôsob reprezentácie používateľského prostredia pomocou pracovnej stránky (*Worklist Service*).

V Sonic ESB Product Family 7.5 vydanej v apríli 2007 pribudol Sonic BPEL Server umožňujúci orchestráciu pomocou štandardného jazyka BPEL. V tejto práci však bola testovaná verzia 7.0, ktorá poskytovala orchestráciu len pomocou produktu Sonic Orchestration Server.

4.2.4 Realizácia požiadaviek

Rôznorodosť systémov a prepojení

Sonic ESB ponúka v rámci produktu Adapters for Sonic ESB cca 300 typov adaptérov pre 35 platforiem, od balíkových (*packaged*) aplikácií (napr. SAP R/3), cez B2B (napr. BizTalk), transakčné, správové (napr. Microsoft MSMQ), CORBA, e-mailové, komunikačné, XML (napr. xCBL), adaptéry pre procedurálne jazyky (napr. PERL, PL/SQL) až po adaptéry na terminály, mainframy a zastarané aplikácie (napr. DBASE, FoxPro). Kompletný zoznam adaptérov sa dá nájsť v [SONDOC].

Medzi existujúce služby dodávané so Sonic ESB patrí smerovanie na základe obsahu správ (CBR), práca so súbormi, služby delenie a spájanie správ a transformácie XML. Taktiež sú tu služby na priamu prácu s webovými službami a službami využívajúcimi JMS - vyvolanie služby pomocou správy SOAP alebo JMS môže byť krok v procese, rovnako ako napríklad vyvolanie služby ESB, odpadá nutnosť použiť nejakú vyvolávaciu „medzislužbu“. Služby pre prácu s databázou, orchestračná služba, či služba na vyhľadávanie procesov sa dodávajú ako samostatné produkty, napr. Sonic Database Service alebo Sonic Orchestration Server.

Maximálna odporúčaná veľkosť správy 10 MB, podpora prenosu väčších správ je realizovaná pomocou streamovacieho kanála typu *peer-to-peer* medzi službami ESB využívajúcimi SonicMQ bez zaťaženia správových sprostredkovateľov.

Robustnosť, dostupnosť, odozva, súčasné prístupy

Škálovateľnosť na úrovni MOM, tj. produktu SonicMQ je vyriešená pomocou *clusteringu*, pri ktorom sa niekoľko fyzických sprostredkovateľov správa ako jeden virtuálny sprostredkovateľ. Pripojenia na virtuálneho sprostredkovateľa sú distribuované medzi fyzických sprostredkovateľov (*load balancing*) rovnomerne alebo na základe definovanej váhy sprostredkovateľa. Existuje viacero vstupných bodov do virtuálne sprostredkovateľa, aby sa predišlo problému bodu zlyhania (*single point of failure*). Menný priestor pre témy (*topics*) a fronty (*queues*) je naviazaný na virtuálneho sprostredkovateľa namiesto fyzického, čím sa docieľuje lepšie vyváženie záťaže - napríklad pri distribuovaných frontách má každý fyzický sprostredkovateľ fragment celkovej distribuovanej fronty s ktorým pracuje a v prípade jeho vyčerpania sa fronta automaticky doplní z iného fyzického sprostredkovateľa.

Jednotliví fyzickí sprostedkovatelia však pri clustrovanom riešení musia byť relatívne pevne previazaní a prepojení, čo nie je vždy vyhovujúce riešenie, napríklad ak sieť

využívajúca túto správovú vrstvu spája viacero geograficky odlišných lokácií, viacero spoločností, prechádza cez pomalé alebo drahé WAN spojenia a podobne. SonicMQ rieši tento problém svojou technológiou nazvanou *Dynamic Routing Architecture*, pri ktorej sú jednotliví sprostredkovatelia (fyzickí alebo virtuálni) prepojení voľnejšie. Spojenia medzi nimi môžu byť nadväzované a rušené dynamicky podľa potreby, autentifikované či s vyvažovaním záťaže. Pri prerušení spojenia medzi sprostredkovateľmi sú správy ukladané a poslané pri obnovení (*store and forward*).

Sonic ESB taktiež umožňuje *clustering* na úrovni služieb. Na zbernicu môže byť pripojených viacero inštancií rovnakej služby, každá inštancia však musí byť nasadená v inom kontajneri. Inštancie zdieľajú spoločnú konfiguráciu v adresárovej službe, majú teda identické správanie.

Pri dostupnosti ťaží Sonic ESB zo svojej *Continuous Availability Architecture* (CAA), založenej na replikácii konfigurácií, kontajnerov a služieb na záložných sprostredkovateľov, pomocou čoho zaručuje *failover*. Pri výskyte chyby na strane klienta alebo sprostredkovateľa sa komunikácia automaticky prepojí na záložného sprostredkovateľa. CAA zaručuje, že záložní sprostredkovatelia budú mať ten istý stav a tú istú *session*.

Bezpečnosť, autentifikácia, autorizácia

Definovateľná bezpečnostná politika SonicMQ rieši bezpečnosť na úrovni správ, na úrovni spojení (SSL alebo iné šifrovanie), autentifikáciu a autorizáciu prístupu ku komunikačnej zbernici, autorizáciu na typ správ. Je umožnená spolupráca s firewallmi od tretích strán, externá autentifikácia (napr. LDAP), Java Authentication and Authorisation Services (JAAS). Autorizácia je riešená pomocou Access Control List, kde sa dajú definovať prístupy na jednotlivé zdroje, napr. JMS adresy.

Sonic ESB implementuje špecifikácie WS-Security, WS-ReliableMessaging, a WS-Policy. WS-ReliableMessaging sa zaoberá doručením správ práve raz a v správnom poradí, WS-Security poskytuje škálovateľnejšie riešenie oproti HTTPS, ktoré napríklad podporuje len jeden *user credential* na connection a tým spôsobuje problémy pri prepájaní viacerých serverov s viacerými *user credentials*. Taktiež zaručuje integritu a dôvernosť na úrovni správ (HTTPS toto zaručuje na úrovni transportnej vrstvy), čiže v rámci jedného spojenia môže prebiehať komunikácia s viacerými úrovňami zabezpečenia. WS-Security tiež špecifikuje ako sa kódujú X.509 certifikáty a Kerberos tickety (X.509 je protokol pre Public Key Infrastructure). WS-Policy spolu s WSDL definuje podmienky, za ktorých môžu byť používané služby, napríklad požadované zabezpečenie alebo spoľahlivosť.

Správa, monitorovanie, logging, auditing

Sonic ESB ponúka niekoľko nástrojov na správu, napríklad Sonic Management Console alebo ESB Admin Tool. Nástroje zabezpečujú konfiguráciu a správu kontajnerov a komponentov, projektov, koncových bodov, služieb, procesov, zdrojov, správovej vrstvy atď. Ďalej zabezpečujú správu životných cyklov, nasadzovanie a monitorovanie komponentov - definovanie metrík a grafov, *logovanie* na úrovni správ (tj. produktu SonicMQ) alebo pri použití Sonic XML Servera na úrovni obsahu správ XML.

4.2.5 Výhody a nevýhody

Výhody

Medzi výhody produktu Sonic ESB patrí kvalitný vývojársky nástroj Workbench, založený na platforme Eclipse, určený na modelovanie, konfigurovanie, testovanie a odlaďovanie lokálnych aj distribuovaných procesov. Ďalej je to množstvo existujúcich adaptérov, či kvalitná dokumentácia. Samozrejmosťou je aj dostupná komerčná podpora.

Nevýhody

Nespornou nevýhodou oproti open source riešeniam je cena a licenčná politika s neprístupným zdrojovým kódom. Vzhľadom na pomerne vysokú komplexnosť produktu Sonic ESB tento vyžaduje nemalý čas na naučenie, či získanie nových schopností, napr. na konfiguráciu a správu. Sonic ESB používa vlastné nástroje a niekedy aj vlastné formáty (napr. procesný jazyk), teda možnosť opakovaného použitia znalostí v iných produktoch je limitovaná.

4.3 Open ESB

Open ESB je relatívne mladý open source produkt (začiatok projektu bol v máji 2006 [FRY06]), ktorý je sponzorovaný spoločnosťou Sun Microsystems a dodávaný pod licenciou CDDL[11]. Open ESB implementuje špecifikáciu JBI (viď kapitola 3.7 Java Business Integration) a pomocou dodávaných prepájacích komponentov (*Binding Component*) a službových prostriedkov (*Service Engines*) poskytuje podporu pre procesnú orchestráciu v jazyku BPEL, prepojenie na externé webové služby pomocou HTTP SOAP a podobne. Open ESB využíva komponenty vyvíjané v rámci projektu Open JBI Components, viac v kapitole 4.3.2 Úroveň služieb.

Open ESB je napísaný v jazyku Java. V čase písania tejto práce je aktuálna verzia Open ESB 2.0 Beta, ktorá je pribalená ako súčasť Java EE 5 SDK[12], prípadne dostupná na stránke projektu [OPEN] alebo [OPEN2B]. Na svoj beh vyžaduje JDK 1.5. Medzi podporované platformy patria MS Windows, Linux a Solaris.

V januári 2007 sme testovali vtedy dostupnú verziu Open ESB 1.0. Pri tejto verzii sa vyskytli mierne problémy pri inštalovaní a testovaní zapríčinené nekonzistentnou dokumentáciou, ani vo verzii 2.0 sa však dokumentácia nedá považovať za dostatočnú. Informácie v tejto kapitole sú preto čerpané jednak z dokumentácie k projektu ([OPEN], [OPEN2B]), ale taktiež z príbuzných zdrojov ako [GOP06], komunitného wiki pre JBI, vývoj JBI Components a Open ESB [JBIWIK] alebo vývojárskeho fóra JBI [JBIFOR].

Open ESB využíva na svoj beh aplikačný server Glassfish alebo Sun Java System Application Server (Glassfish je označenie open source distribúcie aplikačného

[11] http://www.sun.com/cddl/cddl.html

[12] http://java.sun.com/javaee/

servera Sun Java System Application Server) a existuje aj experimentálna verzia Open ESB pre JBoss, plánovaná je podpora pre IBM WebSphere. Ako neskôr uvidíme pri produktoch Apache ServiceMix a Mule, takáto závislosť na jednom aplikačnom serveri je pri open source produktoch skôr ojedinelá.

Zbernica služieb je realizovaná ako jedna alebo viacero prepojených inštancií kontajnerov JBI, každá inštancia JBI beží v aplikačnom serveri. V čase testovania bola podporovaná len jedna inštancia kontajnera JBI v jednom aplikačnom serveri, teda napríklad škálovateľnosť bolo nutné riešiť clustrovaním na úrovni aplikačného servera, nie na úrovni inštancií JBI. Zatiaľ teda nemožno hovoriť o skutočne odľahčenom (*lightweight*) kontajneri.

Preferované vývojárske prostredie nie je na rozdiel od ostatných analyzovaných produktov platforma Eclipse, ale platforma NetBeans, ktorá vo svojej verzii Enterprise Pack 5.5.1 obsahuje nástroje na vytváranie, nasadzovanie a správu aplikácií využívajúcich Open ESB. NetBeans poskytujú grafický editor pre WSDL, nástroje na prácu s schémami XML a XSLT *stylesheets*, editor a *debugger* pre BPEL, nástroje na dizajn a konfigurovanie kompozitných aplikácií - prepájanie koncových bodov služieb, konfigurácie udalostí a nástroje na testovanie a ladenie (*debugging*) služieb. Ďalšie nástroje poskytuje Sun Application Server, napríklad úlohy pre Ant (*Ant tasks*) a príkazové rozhranie (*command line interface*) na správu pomocou príkazového procesora Sun Application Server. Oproti Sonic ESB teda ide o menšiu ponuku nástrojov, oproti ostatným dvom open source produktom (Apache ServiceMix a Mule) sú tieto nástroje však jednoznačnou výhodou.

4.3.1 Úroveň správ

Formát a štruktúra správ

Open ESB implementuje špecifikáciu JBI, teda na úrovni správ má črty opísané v kapitole 3.7 Java Business Integration.

Realizácia komunikačnej zbernice

Funkcionalitu posielania správ poskytuje JMS Binding Component, ktorý vo verzii Open ESB 1.0 spolupracoval len so správovým sprostredkovateľom Sun Java System MQ (SJSMQ).

SJSMQ je samostatne predávaný komerčný produkt, ku ktorému ale existuje aj verzia na voľné stiahnutie, ktorá je pribalená k Sun Java System Application Server. Aktuálna je verzia 3.6 SP 3 z roku 2005. Sun Java System MQ je podnikový (*enterprise-grade*) správový poskytovateľ, porovnateľný s SonicMQ a ponúka riešenie problému dostupnosti, spoľahlivosti a bezpečnosti (viac napríklad v [SJSMQ] alebo v kapitole 4.3.4 Realizácia požiadaviek).

4.3.2 Úroveň služieb

Služba, koncový bod a službový kontajner

Open ESB implementuje špecifikáciu JBI, teda aj na úrovni služieb, koncových bodov a službových kontajnerov má črty popísané v kapitole 3.7 Java Business Integration. Ako už bolo povedané, Open ESB využíva službové prostriedky (S*ervice Engines*) a prepájacie komponenty (*Binding Components*) vyvíjané v rámci projektu Open JBI Components. Nájdeme tu komponenty ako:

Službové prostriedky (*Service Engines*)	**Prepájacie komponenty** (*Binding Components*)
BPEL SE	CORBA BC
Java EE SE	File BC
SQL SE	FTP BC
JavaScript SE (plánované)	HTTP BC
	JDBC BC
	MQ Series BC
	SAP BC
	SMTP BC

Tabuľka 1 - Niektoré komponenty z projektu Open JBI Components

Viac v kapitole 4.3.4 Realizácia požiadaviek.

V Open ESB je možné použiť aj komponenty JBI z iných projektov, napríklad komponenty pôvodne vyvíjané pre Apache ServiceMix. Aj toto je jedna z výhod špecifikácie JBI. Treba však podotknúť, že nie všetky komponenty z Apache ServiceMix sú štandardné komponenty JBI, nakoľko ServiceMix podporuje aj odľahčené komponenty, tzv. JBI POJO (viď 4.4 Apache ServiceMix).

Konfigurácia a správa

Jedna z inštancií kontajnerov JBI je centrálny spravovací server (*Central Administration Server - CAS*), pomocou ktorého sa celá zbernica spravuje a s ktorým komunikujú ostatné inštancie JBI. CAS udržuje centrálny adresár konfigurácií všetkých kontajnerov a komponentov. CAS nie je nevyhnutný pre beh ESB, len pre konfiguráciu, nepredstavuje teda bod zlyhania (*single point of failure*).

4.3.3 Úroveň procesov

Choreografia

Choreografia nie je v JBI podporovaná.

Orchestrácia

O procesné riadenie sa stará službový prostriedok BPEL SE, ktorý implementuje špecifikáciu BPEL (konkrétne WS-BPEL 2.0). Podobný komponent je použitý aj v produkte Apache ServiceMix (4.4.3 Úroveň procesov).

BPEL je jazyk na modelovanie dlhotrvajúcich stavových orchestračných procesov, v ktorých sú prepájané služby definované pomocou WSDL. Verzie jazyka BPEL sú označované ako BPEL4WS 1.0 resp. 1.1 alebo WS-BPEL 2.0 a sú schvaľované organizáciou pre štandardy OASIS[13]. BPEL popisuje procesy pomocou jazyka XML.

BPEL SE podporuje komunikáciu typu požiadavka/odpoveď, asynchrónne jednosmerné vyvolávanie a priame volanie medzi dvoma podprocesmi. Ponúka funkcionalitu na monitorovanie stavu koncových bodov, na vytváranie (*build*) a testovanie nasadených procesov. Viac sa komponentu BPEL SE venuje napríklad zdroj [GOP06].

Open ESB ponúka možnosť vývoja a nasadzovania procesov BPEL pomocou konfiguračných a spravovacích nástrojov NetBeans. Príkladom môže byť grafický nástroj na tvorbu BPEL procesov - BPEL Visual Designer.

4.3.4 Realizácia požiadaviek

Rôznorodosť systémov a prepojení

Open ESB využíva komponenty z projektu Open JBI Components. Nájdeme tu 7 službových prostriedkov (*Service Engines*), napr. Java EE, SQL, XSLT, podpora pre skriptovacie jazyky alebo BPEL 2.0 a 22 prepájacích komponentov (*Binding Components*), napr. pre CORBA, DCOM, File, FTP, HTTP, JMS, JDBC, WebSphere MQ, SAP, SMTP, UDDI. Ďalšie SE a BC sú naplánované. Počet dodávaných komponentov je síce menší ako napríklad pri produkte Apache ServiceMix, avšak kvalitou tieto konkurenčné komponenty čiastočne predbehnú.

Sun Java System Message Queue používaný ako vrstva MOM ponúka možnosť konfigurovateľnej veľkosti správ.

Robustnosť, dostupnosť, odozva, súčasné prístupy

Open ESB poskytuje podporu pre *clustering* na základe podpory pre *clustering* na úrovni aplikačného servera Glassfish, resp. Sun Java System Application Server. Každá inštancia aplikačného servera štandardne obsahuje kontajner JBI, služba

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

nasadená na *cluster* je automaticky distribuovaná do všetkých kontajnerov JBI. S podobným riešením sa môžme stretnúť aj v produkte Apache ServiceMix.

Z administračného pohľadu sa napr. všetky inštancie služby dajú konfigurovať z jedného miesta (CAS spomínaný v kapitole 4.3 Open ESB) a jednotným rozhraním. Interne si však komponenty musia *clustering* a *load balancing* musia riešiť sami. Napríklad BPEL SE, ktorý má na starosti koreláciu požiadaviek a odpovedí musí vedieť identifikovať správnu inštanciu cieľovej služby, napríklad na základe nejakého identifikátora (*correlation ID*). Funkcionalita *clusteringu* komponentov je do veľkej miery závislá na službách ktoré poskytujú a spôsobe, akým komponenty komunikujú. Napríklad HTTP BC sa môže pri vyvažovaní záťaže spoľahnúť na externý *load balancer*, ktorý bude distribuovať požiadavky rovnomerne medzi jednotlivé inštancie tohto komponentu, keďže všetky inštancie daného HTTP BC majú rovnakú konfiguráciu.

Na úrovni MOM ponúka predvolený Sun Java System Message Queue podporu pre *clustering* správových sprostredkovateľov, ktorí sú prepojení v topológii kompletného grafu, rozdeľujú si záťaž, navzájom sa synchronizujú a replikujú medzi sebou informácie, napríklad o cieľoch JMS (*JMS destination*).

Bezpečnosť, autentifikácia, autorizácia

V Open ESB existuje podpora pre WS-I Basic Security Profile z webových služieb. Tento sa zaoberá bezpečnosťou na úrovni SOAP správ a na úrovni prenosov (HTTPS). Okrem toho je v Open ESB už len podpora základnej autentifikácie (*basic authentication*) pomocou nezašifrovaného mena a hesla v komponente HTTP BC. HTTP BC zatiaľ nepodporuje špecifikácie WS-Security ani WS-Rel*. Niektoré komponenty (napr. SMTP BC) majú problémy už s HTTPS prenosmi.

Celkovo možno povedať, že bezpečnostná funkcionalita Open ESB je zle zdokumentovaná. Chýba celkový prehľad, napríklad spôsob prepojenia so Sun Java System Access Manager, čo je produkt určený na autentifikáciu a autorizáciu a spolu s Open ESB je dodávaný v balíku Java Application Platform SDK. Komponent LDAP BC ešte nie je vyvinutý, len naplánovaný.

Správa, monitorovanie, logging, auditing

Nástroje na správu a monitorovanie sú sučasťou balíka NetBeans. Bližšie sú popísané v kapitole 4.3 Open ESB.

Na zaznamenávanie (*logging*) sa používa funkcionalita aplikačného servera Sun Java System Application Server s konfigurovateľnými výstupmi.

4.3.5 Výhody a nevýhody

Výhody

Medzi výhody patria cena, nakoľko Open ESB je dostupné zadarmo a dostupné nástroje na administráciu, ktoré sú súčasťou prostredia NetBeans.

Nevýhody

Vážnymi nedostatkami sú dostupnosť a kvalita dokumentácie, naviazanosť na jeden aplikačný server a nedostatočná poskytovaná funkcionalita, najmä v oblasti bezpečnosti.

4.4 Apache ServiceMix

Apache ServiceMix je ďalší open source produkt typu ESB implementujúci špecifikáciu JBI (viď kapitola 3.7 Java Business Integration). Projekt bol spustený v auguste 2005 a spočiatku bol ServiceMix sponzorovaný spoločnosťou LogicBlaze, ktorá na ňom postavila svoju platformu Fuse SOA ([FRY06]). V čase písania tejto práce bola aktuálna verzia Apache ServiceMix 3.1 z februára 2007, ktorú sme aj testovali. Projekt je stále „umiestnený“ v Apache Incubator[14], teda projekt je ešte oficiálne vo fáze vývoja. Informácie v tejto kapitole sú čerpané prevažne z dokumentácie dostupnej na stránke projektu [SMIX], ktorá poskytuje relatívne kvalitnú dokumentáciu, aj keď nie vždy aktualizovanú pre poslednú verziu.

Apache ServiceMix je vydávaný pod licenciou Apache 2.0[15], v prípade potreby poskytuje spoločnosť LogicBlaze komerčnú podporu.

ServiceMix je napísaný v programovacom jazyku Java, na beh vyžaduje JDK 1.4 alebo vyššie. Podporuje platformy MS Windows, Mac OS X, Solaris a Linux. Niektoré komponenty (napríklad službový prostriedok pre BPEL) môžu vyžadovať JDK 1.5 alebo vyššie.

ServiceMix poskytuje open source implementáciu odľahčeného kontajnera JBI. Kontajner môže byť nasadený samostatne, ako komponent do iného kontajnera JBI alebo ako WAR v kontajneroch J2EE (Apache Geronimo, BEA Weblogic, RedHat JBoss, Glassfish/SJSAS, ...). Naproti tomu napr. Open ESB je zatiaľ naviazané na Glassfish/SJSAS a existuje len limitovaná podpora JBoss.

ServiceMix podporuje Spring Framework, ktorý môže byť využitý pre nasadzovanie a konfiguráciu a poskytuje podporu pre správu projektov pomocou Apache Maven. ServiceMix podporuje správu kontajnerov a komponentov pomocou nástrojov podporujúcich štandard JMX. Ako vývojárske rozhranie je preferovaná platforma Eclipse. Žiaľ, v porovnaní s Open ESB chýbajú nástroje uľahčujúce vývoj, ako napr. editor pre WSDL. Konfigurácia kontajnerov, služieb a pod. je vyriešená pomocou priamej editácie súborov XML.

[14] http://incubator.apache.org/

[15] http://www.apache.org/licenses/LICENSE-2.0.html

4.4.1 Úroveň správ

Formát a štruktúra správ

Apache ServiceMix implementuje špecifikáciu JBI, teda na úrovni správ má črty opísané v kapitole 3.7 Java Business Integration. Pri definovaní smerovacích pravidiel ponúka podporu skriptovacích jazykov založených na špecifikácii JSR 223 – napríklad JavaScript, Jython alebo Groovy.

Realizácia komunikačnej zbernice

Apache ServiceMix využíva na správovej vrstve open source produkt Apache ActiveMQ, čo je správový poskytovateľ podporujúci špecifikáciu JMS 1.1, rieši otázky bezpečnosti (SSL, JAAS, ...), spoľahlivosti a dostupnosti (*clustering*, replikovanie správ medzi sprostredkovateľmi). Viac v kapitole 4.4.4 Realizácia požiadaviek. Aktuálna verzia ActiveMQ je 4.1.1 z apríla 2007, ide teda o stále aktualizovaný a udržovaný produkt.

4.4.2 Úroveň služieb

Služba, koncový bod a službový kontajner

Apache ServiceMix implementuje špecifikáciu JBI, teda aj na úrovni služieb, koncových bodov a službových kontajnerov má črty popísané v kapitole 3.7 Java Business Integration. ServiceMix poskytuje implementáciu odľahčeného kontajnera JBI. Súčasťou projektu je aj približne 40 službových prostriedkov a prepájacích komponentov, viac v kapitole 4.4.4 Realizácia požiadaviek.

Nakoľko Apache ServiceMix implementuje špecifikáciu JBI, je možné v ňom použiť aj komponenty vyhovujúce špecifikácii JBI od iných dodávateľov, napríklad komponenty vyvíjané v rámci projektu Open JBI Components, ktoré využíva projekt Open ESB (viac v kapitole 4.3 Open ESB). Niektoré z komponentov v projekte ServiceMix však nie sú plnohodnotné komponenty JBI, nakoľko ServiceMix podporuje aj odľahčené verzie komponentov. Preto je možné, že niektoré komponenty z projektu ServiceMix nebude možné použiť v iných implementáciách JBI.

ServiceMix okrem tradičných komponentov JBI aj odľahčenú verziu komponentov, ktoré sa nazývajú *JBI POJO* (*Plain Old Java Object*). Od takýchto komponentov sa vyžaduje iba aby poskytli rozhranie na konfiguráciu a správu životného cyklu, sú teda jednoduchšie ako typické komponenty JBI. Podľa špecifikácie JBI sa totiž od každého komponentu JBI vyžaduje okrem implementácie rozhrania na ovládanie životného cyklu napríklad to, dokázal pracovať s metadátami alebo poskytoval rozhranie na správu komponentu, čím sa zabezpečí medzi komponentmi vzájomná kompatibilita. Niekedy je však toto zbytočná záťaž pri dizajne a vývoji a komponenty JBI nie sú také odľahčené, ako by mohli byť - okrem naozaj nutnej funkcionality musia obsahovať aj funkcionalitu požadovanú od prostredia JBI.

Konfigurácia a správa

V Apache ServiceMix používa každá inštancia kontajnera JBI vlastné konfigurácie vo forme súborov XML. Tieto konfigurácie sú sprístupnené pomocou rozhrania JMX, prostredníctvom ktorého sa tiež ovláda životný cyklus komponentov, či monitoruje kontajner. Na rozdiel od ostatných analyzovaných produktov teda v ServiceMix neexistuje centrálne miesto na správu konfigurácií.

4.4.3 Úroveň procesov

Choreografia

Choreografia nie je v JBI podporovaná.

Orchestrácia

ServiceMix, podobne ako produkt Open ESB využíva na orchestráciu komponent pracujúci s procesným jazykom WS-BPEL 2.0 (viac o jazyku BPEL v kapitole 4.3.3 Úroveň procesov). ServiceMix využíva externý komponent PXE of spoločnosti FiveSight[16], ktorý sa v roku 2006 transformoval na projekt Apache ODE. PXE/ODE podporuje dlhotrvajúce transakcie a stavové procesy. Vďaka architektúre JBI dokáže prepájať ľubovoľnú službu definovanú pomocou WSDL. Avšak kvôli tomu, že ServiceMix povoľuje odľahčené komponenty JBI, ktoré plne nevyhovujú špecifikácii JBI, občas môžu nastať problémy pri zapájaní takýchto komponentov do BPEL procesov.

4.4.4 Realizácia požiadaviek

Rôznorodosť systémov a prepojení

ServiceMix obsahuje približne 40 komponentov - interných, ktoré sú súčasťou projektu, ale aj externých od iných dodávateľov. Medzi službové prostriedky (SE) patria napríklad: smerovanie na základe obsahu (CBR) správ XML, časovač, podpora komponentov EJB, podpora skriptovacích jazykov, transformácie a validácie XML, procesná orchestrácia pomocou jazyka BPEL, implementácia WS-Notification. Medzi prepájacie komponenty (BC) patria napríklad: práca so súborovým systémom, FTP, Email, CORBA, práca s databázou pomocou JDBC, prepojenie s webovými službami cez HTTP/HTTPS a SOAP alebo prepojenie s JMS.

Produkt ActiveMQ používaný v správovej vrstve podporuje priamy prenos teoreticky ľubovoľne veľkých správ pomocou JMS Stream. Pri veľkých správach môže byť uprednostnený prenos typu *peer-to-peer* medzi producentom a konzumentom bez prechodu správovými sprostredkovateľmi pomocou formátu Blob Message.

Robustnosť, dostupnosť, odozva, súčasné prístupy

[16] http://pxe.fivesight.com/

Na úrovni správ využíva Apache ServiceMix produkt Apache ActiveMQ, ktorý zabezpečuje spoľahlivosť a dostupnosť tým, že poskytuje funkcionalitu pre *clustering* a distribuovaný *failover* správových sprostredkovateľov.

Na úrovni služieb zabezpečuje realizáciu týchto požiadaviek ServiceMix pracujúci v *clustered* móde tým, že využíva funkcionalitu ActiveMQ na automatické objavenie (*auto discovery*) ostatných inštancií kontajnerov ServiceMix, komponentov JBI a koncových bodov. Podobným spôsobom pracuje aj Open ESB.

Bezpečnosť, autentifikácia, autorizácia

Na úrovni prenosu správ podporuje produkt ActiveMQ prenosy zabezpečené pomocou SSL alebo WS-Security, v čase písania tejto práce bola implementácia WS-Security len čiastočná. Autentifikácia a autorizácia používateľov prebieha pomocou JAAS ale je podporovaná aj základná autentifikácia (*basic authentication*) pomocou nezašifrovaného mena a hesla v prepájacom komponente servicemix-http.

Nevýhodou je, že v Apache ServiceMix zatiaľ neexistuje podpora pre LDAP, Kerberos alebo protokoly WS-Rel*. Celkovo je bezpečnostná politika slabo zdokumentovaná.

Správa, monitorovanie, logging, auditing

Spravovanie, monitorovanie a audit kontajnerov a komponentov (služieb) je možné prostredníctvom štandardných nástrojov podporujúcich špecifikáciu JMX, napríklad nástroja JConsole, ktorý je dodávaný spolu s J2SE 5.0.

Na zaznamenávanie (*logging*) sa používa štandardný nástroj log4j s konfigurovateľnými výstupmi.

4.4.5 Výhody a nevýhody

Výhody

Medzi výhody Apache ServiceMix sa radí cena, nakoľko je tento produkt dostupný zadarmo, podpora pre ľubovoľný kontajner J2EE a dobrá integrácia s ostatnými produktmi Apache Software Foundation a inými open source produktmi.

Nevýhody

ServiceMix má menej grafických nástrojov (editory, konfiguračné nástroje) oproti Open ESB, väčšina konfigurácií sa uskutočňuje priamou editáciou XML súborov.

4.5 Mule

Mule je produkt typu ESB založený na princípoch SEDA (*Staged Event-Driven Architecture*)[17]. SEDA sa dá chápať ako dizajnový vzor na vytváranie vysoko

[17] http://www.eecs.harvard.edu/~mdw/proj/seda/

výkonných a škálovateľných distribuovaných systémov, založený na dekompozícii komplexného udalosťami riadeného systému na množinu úrovní (*stage*) prepojených frontami (*queue*). Architektúre SEDA sa hlbšie venujeme v kapitole 4.5.4 Realizácia požiadaviek. Projekt Mule bol spustený v decembri 2004 ([FRY06]) a je dostupný pod open source licenciou MPL+ odvodenou od Mozilla Public Licence. Komerčnú podporu poskytuje spoločnosť MuleSource[18].

Informácie v tejto kapitole sú čerpané zo stránky Mule [MULE], ktorá poskytuje v porovnaní s Open ESB alebo Apache ServiceMix relatívne kvalitnú dokumentáciu.

Mule je napísaný v programovacom jazyku Java, na beh vyžaduje JDK 1.4 alebo vyššie. Podporované sú platformy MS Windows, Mac OS X, Solaris, AIX, HP-UX a Linux.

V čase písania tejto práce bola aktuálna verzia 1.4. Mule môže bežať samostatne ako Mule Server, alebo byť nasadený v niekoľkých aplikačných serveroch: Apache Geronimo, JBoss, Oracle Application Server, IBM WebSphere, BEA Weblogic alebo Glassfish/Sun Java System Application Server. Mule Server môže byť vsadený (*embedded*) do inej aplikácie Java alebo webovej aplikácie Java.

Komponenty Mule môžu byť nasadené a spolupracovať s kontajnerom JBI a ostatnými komponentmi JBI. Celkovo sa však dá povedať, že aj keď sa Mule a JBI zaoberajú rovnakou doménou, venujú sa jej z inej perspektívy. Špecifikácia JBI je viac orientovaná na štandardy XML a WSDL, napríklad vyžaduje aby správy boli vo formáte XML. Vo všeobecnosti definuje JBI oproti Mule striktnejšie ohraničenia pre komponenty, správy a podobne. Vo vývoji je však projekt označovaný ako Mule-JBI, ktorý poskytne implementáciu kontajneru JBI s využitím *frameworku* Mule. Špecifikácii JBI sa viac venujeme v kapitole 3.7 Java Business Integration.

Mule sa na ESB pozerá ako na topológiu s jednou zbernicou. Okrem tejto topológie podporuje aj ďalšie, napríklad *pipeline*, *peer network*, *client/server* alebo *hub-and-spoke*. Mule je inšpirovaný publikáciou Enterprise Integration Patterns [HOH04], z ktorej prevzal aj časť terminológie, napríklad pri smerovačoch (*routers*), ktorým sa venujeme v kapitole 4.5.1 Úroveň správ.

Podobne ako Apache ServiceMix, aj Mule podporuje Spring Framework, ktorý môže byť využitý pre nasadzovanie a konfiguráciu, rovnako existuje aj podpora pre správu projektov pomocou Apache Maven. Mule navyše ponúka vývojárske prostredie Mule IDE, založené na platforme Eclipse. Mule IDE je stále vo fáze vývoja, časť funkcionality je ešte len naplánovaná alebo vo vývoji, napríklad grafický nástroj na editáciu konfigurácií - momentálne sa konfigurácie ručne píšu do XML súborov.

[18] http://www.mulesource.com/

4.5.1 Úroveň správ

Formát a štruktúra správ

Mule má v podstate veľmi voľné kritériá na správy, napríklad v porovnaní s JBI, ktoré vyžaduje správy vo formáte XML s určitou normalizovanou štruktúrou (viď kapitola 3.7 Java Business Integration). Mule podporuje správy vo formáte reťazcov, XML, serializovateľných objektov, ľubovoľných binárnych tokov (*binary stream*) a podobne. Konkrétny formát a štruktúra správy závisí od konfigurácie poskytovateľa prenosu (*transport provider*), resp. konektora (*connector*), z ktorého prichádza správa, resp. do ktorého správa odchádza zo služby. Poskytovatelia prenosu, resp. konektory sú napojené na koncový bod služby. Viac v nasledujúcej kapitole. Telo správy tvorí časť prijatých dát (napríklad telo správy JMS, alebo telo požiadavky HTTP), ostatné dáta sú prenášané so správou a dostupné ako vlastnosti správy (*properties*). Vlastnosti sú napr. položky hlavičky (napríklad priorita správy JMS) alebo metadáta (názov súboru pri poskytovateľovi pre prácu so súborovým systémom alebo atribúty ako *content-type* pri poskytovateľovi pre HTTP). Správy môžu obsahovať prílohy (v prípade, že to protokol, s ktorým pracuje poskytovateľ prenosu, podporuje – napríklad SOAP, SMTP alebo JBI).

Príkladom poskytovateľa môže byť poskytovateľ pre JMS, vďaka ktorému vie Mule spolupracovať s mnohými servermi podporujúcimi špecifikáciu JMS, napríklad ActiveMQ, JBoss MQ, OpenJms, SonicMQ, IBM WebSphere MQ, BEA WeblogicMQ. Zoznam preddefinovaných poskytovateľov prenosu a konektorov je uvedený v kapitole 4.5.4 Realizácia požiadaviek.

Realizácia komunikačnej zbernice

Na rozdiel od ostatných analyzovaných produktov Mule nepoužíva iný existujúci produkt typu MOM na prácu so správami. O posielanie a smerovanie správ sa stará kontajner a koncové body služieb, ktorým sa viac venujeme v nasledujúcej kapitole. Škálovateľnosť takéhoto riešenia zabezpečuje architektúra SEDA, na základe ktorej bol kontajner Mule vyvíjaný. Škálovateľnosti sa hlbšie venujeme v kapitole 4.5.4 Realizácia požiadaviek.

4.5.2 Úroveň služieb

Služby

Služby sa v Mule nazývajú *Universal Message Object Component – UMO Component*, alebo len skrátene UMO. Mule ponúka implementáciu odľahčeného distribuovaného kontajnera (označovaného ako *Mule Model*), v ktorom sú nasadené jednotlivé UMO. UMO komunikujú s inými UMO a s externými aplikáciami (tj. aplikáciami, ktorá nie sú nasadené v kontajneri Mule) prostredníctvom svojich koncových bodov, ktoré poskytujú abstrakciu nad komunikačným protokolom.

Zaujímavou črtou UMO je, že v podstate ide o bežný komponent JavaBean a neobsahuje žiaden kód špecifický pre Mule. Tento prístup, keď sú na komponent kladené minimálne nároky, je podobný odľahčeným komponentom JBI POJO pri Apache ServiceMix, ktoré sú popísané v kapitole 4.4.2 Úroveň služieb.

Koncový bod

Koncový bod je napojený na niektorého poskytovateľa prenosu, napríklad poskytovateľa pre JMS. Správy môžu byť na koncovom bode transformované, filtrované a smerované. Smerovanie správ odchádzajúcich a prichádzajúcich do koncových bodov má za úlohu objekt s názvom smerovač (*router*). Smerovače ponúkajú rôznu funkcionalitu, napríklad rozdeľovanie/agregácia správ, transformácia správ, zmena poradia správ alebo vylúčenie duplicitných správ. Smerovače môžu byť ďalej prepojené s filtrami, napríklad filtrami na obsah správy. Preddefinované smerovače sú inšpirované publikáciou [HOH04].

Koncové body sú pomocou smerovačov prepojené do „kanálov" (*channels*), napríklad výstupný koncový bod jednej služby môže byť prepojený pomocou troch kanálov na tri rôzne vstupné koncové body iných služieb. Po ktorom kanáli bude odchádzajúca správa smerovaná, závisí od nastavenia filtra, resp. smerovača, nie od samotnej služby. Výstupné správy môžu byť smerované podľa obsahu (CBR), podľa vlastností správy (*properties*) alebo podľa itinerára uloženého v správe (itinerárové smerovanie je popísané v kapitole 3.4.3 Úroveň procesov). V Mule je teda úloha smerovania správ posunutá na koncový bod, v ostatných analyzovaných produktoch môže samotná služba určiť adresu cieľovej služby, resp. jej koncového bodu.

Služobný kontajner

Tok správ medzi jednotlivými koncovými bodmi služieb riadi kontajner (*Mule Model*), ktorý taktiež kontroluje spravovanie vlákien (*thread management*), ovládanie životných cyklov UMO, správu transakcií, logging, auditing a celkovú správu UMO. Okrem prenosu správ medzi koncovými bodmi služieb nasadených v kontajneri sa kontajner stará aj o prenos správ do koncových bodov služieb nasadených v iných kontajneroch. Z tohto pohľadu poskytujú kontajnery v Mule funkcionalitu, ktorú pri ostatných analyzovaných produktoch poskytovali produkty typu MOM.

Konfigurácia a správa

Všetky kontajnery v jednej inštancii servera spravuje centrálny Mule Manager, prostredníctvom ktorého sú dostupné aj všetky konfigurácie kontajnerov, komponentov, poskytovateľov prenosu a podobne.

4.5.3 Úroveň procesov

Choreografia

Vďaka črtám architektúry Mule sa riadenie pomocou choreografie implementuje veľmi ľahko. V Mule je väčšina komunikačnej logiky presunutá na kontajner, v konfigurácii koncového bodu služby je možné definovať postupnosť smerovačov správ a filtrov, na základe ktorých budú prichádzajúce a odchádzajúce správy smerované. Ako sme už spomínali v kapitole 4.5.1 Úroveň správ, smerovanie môže prebiehať na základe obsahu správ (CBR), vlastností (*properties*) správ a podobne.

Orchestrácia

Z pohľadu orchestrácie Mule priamo neposkytuje komponent, ktorý by dokázal riadiť komunikáciu medzi ostatnými komponentmi (službami) na základe definovaných procesných pravidiel. Mule však poskytuje konektor pre integráciu s externými systémami pre BPM (*Business Process Management*), ktoré poskytujú Java API, napríklad JBoss jBPM. Pomocou tohto konektora môžu udalosti v Mule spúšťať a ukončovať procesy alebo meniť ich vnútorný stav. Rovnako, procesy zo systému BPM môžu generovať synchrónne a asynchrónne udalosti pre Mule. Systémy BPM, ktoré sú založené na jazyku BPEL je možné integrovať s Mule pomocou štandardných volaní webových služieb.

4.5.4 Realizácia požiadaviek

Rôznorodosť systémov a prepojení

V Mule zabezpečujú napojenie na rôzne komunikačné protokoly, technológie či systémy objekty označované ako poskytovatelia prenosu (*transport providers*) a konektory (*connectors*), napríklad pre EJB, email (SMTP, POP3, IMAP), súborový systém, FTP, HTTP/HTTPS, JDBC, JMS, WSDL, SOAP (pomocou tohto poskytovateľa môžu byť UMO viditeľné ako webové služby), SSL/TLS, TCP/UDP. Celkovo ponúka Mule 24 poskytovateľov prenosu.

Mule podporuje prenos teoreticky ľubovoľne veľkých správ pomocou poskytovateľov prenosu pracujúcich s tokmi (*stream*) (napríklad poskytovateľ pre prácu so súborovým systémom, FTP, HTTP/HTTPS alebo SOAP) alebo s prílohami (napríklad SMTP, SOAP, JBI).

Robustnosť, dostupnosť, odozva, súčasné prístupy

Na rozdiel od ostatných analyzovaných produktov nie je škálovateľnosť v Mule, založená priamo na *clusteringu*, ale na princípoch architektúry SEDA, ktorá bola spomínaná v úvode kapitoly 4.5 Mule. V architektúre SEDA sa systém skladá z prepojených úrovní (*stage*). Každá úroveň sa ďalej skladá zo vstupnej fronty udalostí (*incomming event queue*), skupiny vlákien (*thread pool*), spracovávača udalostí (*event handler*) a ovládača zdrojov (*resource controller*). Spracovávač udalostí vykonáva samotnú aplikačnú logiku - spracuje vstupnú udalosť, prípadne pošle udalosti pre ďalšie úrovne. Veľkosť skupiny vlákien v jednej úrovni je dynamická, môže sa zväčšovať alebo zmenšovať – toto riadi práve ovládač zdrojov podľa potreby úrovne, napríklad na základe veľkosti vstupnej fronty, čím sa docieli optimálne priraďovanie systémových zdrojov (vlákien) pre všetky úrovne. O zväčšení záťaže sú notifikované predchádzajúce úrovne, ktoré posielajú vstupné správy, ale aj spracovávač udalostí, ktorý môže v prípade potreby znížiť úroveň poskytovaných služieb (*service degradation*).

V Mule je architektúra SEDA implementovaná tak, že každá služba nasadená v kontajneri je považovaná za úroveň (*stage*). Vstupná fronta je umiestnená na vstupnom koncovom bode, spracovávač udalostí je samotná služba. Niektoré z úloh ovládača zdrojov sa dajú konfigurovať na úrovni služieb (napríklad maximálny počet vlákien, ktorý môže služba využiť), o iné sa stará kontajner. Štatistiky, napríklad

veľkosť vstupnej fronty, sa dajú použiť pri smerovaní na výstupnom koncovom bode. Okrem združovania vlákien (*thread pooling*), podporuje Mule aj implicitné hromadenie služieb a poskytovateľov prenosu, v prípade potreby sa môžu automaticky vytvárať nové inštancie týchto komponentov. Hromadenie komponentov je konfigurovateľné na úrovni komponentov.

Mule v súčasnosti nepodporuje *clustering* kontajnerov na poskytnutie funkcionality pre *failover*.

Bezpečnosť, autentifikácia, autorizácia

Bezpečnosť je v Mule riešená pomocou Acegi Security System, JAAS, PGP alebo štandardov webových služieb.

Acegi je framework založený na Spring poskytujúci autentifikačné a autorizačné služby pomocou rôznych zapojiteľných bezpečnostných poskytovateľov (*security provider*), napríklad podpora pre *single sign-on*, autentifikácia pomocou JAAS, LDAP alebo X.509 (protokolu pre Public Key Infrastructure) a pod. Pomocou Acegi sa dá riešiť aj bezpečnosť na úrovni volania metód. Autorizácia môže prebiehať na úrovni služieb (komponentov), ale aj operácií v rámci služieb.

Okrem Acegi existuje v Mule aj priama podpora pre autentifikáciu pomocou JAAS, šifrovanie správ pomocou PGP alebo zabezpečenie komunikácie s webovými službami pomocou WS-Security. V čase písania tejto práce neboli v Mule podporované štandardy WS-Rel*. Na úrovni poskytovateľov prenosu môže byť bezpečnosť riešená napríklad pomocou SSL/TLS.

Správa, monitorovanie, logging, auditing

Mule ponúka len málo vlastných nástrojov na vývoj alebo správu, napríklad vývojárske prostredie Mule IDE založené na Eclipse, ktoré je vo fáze vývoja. Na ovládanie a auditing kontajnerov, komponentov a pod. sa dajú použiť nástroje založené na štandarde JMX. Konfigurácia kontajnera, komponentov a podobne sa uskutočňuje prostredníctvom súborov XML.

Na zaznamenávanie (*logging*) sa používa štandardný nástroj log4j s konfigurovateľnými výstupmi.

4.5.5 Výhody a nevýhody

Výhody

Medzi výhody Mule patrí cena, nakoľko tento produkt je dostupný zadarmo. Ďalej je to jeho ľahká naučiteľnosť - konfigurovanie kontajnerov, komponentov a pod. pomocou XML je veľmi priamočiare a zrozumiteľné a súčasne sa s ním dá dosiahnuť široké spektrum funkcionalít od transformácie a smerovania správ až po združovanie vlákien (*thread pooling*) v kontajneri. Ľahká naučiteľnosť je umožnená aj vďaka voľným kritériám pre správy a komponenty (služby).

Mule podporuje integráciu s mnohými produktmi, od aplikačných serverov až po bezpečnostné *frameworky*. Dá sa použiť v rôznych topológiách, pričom ESB je vnímané len ako jedna z možných topológií.

Nevýhody

Prílišná voľnosť kritérií pre správy a komponenty (služby) môže byť zároveň aj nevýhodou. Táto voľnosť spôsobuje, že veľká časť prepájania služieb sa deje prostredníctvom konfigurácie kontajnera, resp. konkrétne koncových bodov daných služieb. Takéto riešenie znižuje interoperabilitu služieb a v prípade nasadenia služieb do ďalšieho kontajnera (napríklad pre potreby vyvažovania záťaže) je treba konfiguráciu kontajnera zduplikovať, čo môže spôsobovať problémy pri udržiavaní. Naproti tomu napríklad pri JBI, kde sa od služieb vyžaduje viac kritérií (komunikujú len pomocou normalizovaných správ vo formáte XML, komunikujú prostredníctvom preddefinovaných MEP (viď kapitola 3.7.1 Úroveň správ) a pod.) sa služby ľahšie prepájajú a vyžadujú menej konfigurácie na strane kontajnera.

Ďalšou nevýhodou je absencia procesného riadenia, v Mule je podporovaná len integrácia s externým procesným prostredím, ktoré produkuje a konzumuje udalosti pre Mule.

4.6 Výkonnostné testy a testy spoľahlivosti

Aby bolo výkonnostné testovanie objektívne, potrebovali by sme zabezpečiť optimálne podmienky pre každý z testovaných produktov. To zahŕňa nastavenia operačného systému (napr. veľkosť virtuálnej pamäte), nastavenia Java VM (veľkosť pamäte, *garbage collecting*, ...), nastavenia samotnej aplikácie a nastavenia podporných aplikácií. Nakoľko nie všetky z týchto nastavení sú uvádzané v dokumentácii ku všetkým produktom a rovnako aj z časových dôvodov sme sa rozhodli od výkonnostného testovania upustiť.

Podobne sme sa z časových dôvodov rozhodli upustiť od testov spoľahlivosti analyzovaných produktov. Tieto testy by však určite mali určite svoje opodstatnenie, hlavne pri open source produktoch, ktoré sú oproti Sonic ESB ešte v relatívne „mladé“.

4.7 Zhrnutie

V tejto kapitole zhrnieme výsledky, ktoré sme získali analýzou a porovnávaním produktov Sonic ESB, Open ESB, Apache ServiceMix a Mule.

4.7.1 Porovnanie spoločných konceptov

V tejto kapitole zhrnieme porovnanie hlavných konceptov architektúry ESB realizovaných v jednotlivých skúmaných produktoch. Medzi tieto koncepty patria správy, služby, koncové body, službové kontajnery, choreografia, orchestrácia a spôsob konfigurácie a správy v danom produkte. Produkty Open ESB a Apache ServiceMix sa v niektorých konceptoch zhodujú, nakoľko obidva produkty implementujú špecifikáciu JBI. V takom prípade je namiesto názvu produktu pri danom koncepte uvedené, že ide o koncept prevzatý z JBI.

Správa

Dominantným formátom správ vo všetkých analyzovaných produktoch je XML. Sonic ESB síce podporuje aj iné formáty správ, XML je však uprednostňovaný. Jedine Mule špecifikuje veľmi voľné kritériá na správy, formát XML nie je vyžadovaný.

V štruktúre správ sú si produkty podobné, správy majú hlavičku a telo. Vo všetkých produktoch môžu správy obsahovať prílohy.

Služba

Vo všetkých produktoch okrem Mule sú rozhrania služieb popísané pomocou WSDL, v Sonic ESB je takýto popis nepovinný. Mule implementuje služby ako bežné objekty Java (POJO) a predpisuje im len minimálne požiadavky, ktoré musia spĺňať, služby nemusia obsahovať žiaden kód špecifický pre Mule. Podobný prístup má aj Apache ServiceMix, ktorý okrem komponentov (služieb) JBI podporuje aj odľahčené komponenty POJO, tieto však musia implementovať rozhranie na správu životného cyklu. Služby v Sonic ESB, alebo služby JBI v Apache ServiceMix a Open ESB sú už viac naviazané na prostredie (musia implementovať niektoré rozhrania, používajú špecifický kód alebo API).

V JBI sa oddeľuje aplikačná logika od komunikačnej pomocou rozdelenia komponentov (služieb) na službové prostriedky a prepájacie komponenty, podobný koncept je aj v Mule, avšak tu je komunikačná logika presunutá na koncový bod. V Sonic ESB rieši časť komunikačnej logiky správová vrstva Sonic MQ.

Koncový bod

Vo všetkých produktoch okrem Mule môže služba pri posielaní správ špecifikovať adresu koncového bodu služby, pre ktorú je správa určená. Ak sa nešpecifikuje, kontajner zariadi smerovanie správ k správnej službe napríklad na základe názvu služby (ak je špecifikovaný), formátu správy alebo podobne. V Mule sa smerovanie správ docieľuje prepájaním koncových bodov v konfigurácii kontajnera podľa určitých kritérií, ktoré správa spĺňa.

Ostatné charakteristiky koncových bodov tak, ako sme ich opísali v kapitole 3.4.2 Úroveň služieb, sú vo všetkých analyzovaných produktoch podobné.

Službový kontajner

Všetky analyzované produkty implementujú službový kontajner s podobnými charakteristikami, ako sme opísali v kapitole 3.4.2 Úroveň služieb.

Produkty ponúkajú rôzne „úrovne odľahčenia“ kontajneru. Kontajner Open ESB je naviazaný na konkrétny aplikačný server. Kontajner v Sonic ESB vyžaduje iba samostatné JVM. Skutočne odľahčený službový kontajner poskytuje Apache ServiceMix a Mule, tieto kontajnery majú najvoľnejšie požiadavky na prostredie, v ktorom sú nasadené, dokážu byť vložené (*embedded*) do inej aplikácie a podobne.

Choreografia

Produkty Sonic ESB a Mule podporujú riadenie komunikácie medzi službami pomocou choreografie, v JBI prebieha komunikácia vždy medzi konzumentskou a producentskou službou, preto nie je možné prenechať úlohu riadenia komunikácie na kontajner, ktorý by smeroval správy zodpovedným službám na základe itinerára obsiahnutého v správach.

Orchestrácia

Produkty Open ESB a Apache ServiceMix využívajú komponenty pracujúce s procesmi špecifikovanými v jazyku BPEL. Naproti tomu Sonic ESB používa vlastný procesný jazyk, komponent pre BPEL je dodávaný až v novšej verzii. Sonic ESB aj Open ESB majú dobrú podporu nástrojov pri ladení (*debugging*) procesov.

Mule nemá priamo komponent, ktorý by bol schopný procesného riadenia, spolieha sa na externé komponenty.

Konfigurácia a správa

Všetky produkty okrem Apache ServiceMix majú centralizovanú správu konfigurácií. Všetky analyzované produkty používajú na správu nástroje pracujúce podľa štandardu JMX, Sonic ESB a Open ESB však ponúkajú vlastné, prípadne štandardné nástroje, ktoré poskytujú veľa funkcionality pri vývoji, odlaďovaní a správe aplikácií.

4.7.2 Zhrnutie vlastností analyzovaných produktov

V tejto kapitole sa zameriame na celkové zhrnutie vlastností analyzovaných produktov.

Všetky analyzované produkty sú napísané v jazyku Java, všetky vyžadujú na svoj beh rovnaké prostredie a tým pádom sú aj podporované platformy veľmi podobné.

Komerčný produkt Sonic ESB poskytuje najkomplexnejšiu funkcionalitu spomedzi všetkých analyzovaných produktov. Toto môže byť do veľkej miery spôsobené tým, že produkt bol vyvíjaný o približne tri až štyri roky dlhšie ako analyzované open source produkty. Na druhej strane však pravdepodobne aj kvôli tomu ešte plne nepodporuje novšie štandardy, ako napríklad BPEL (až v novšej verzii 7.5). Navyše, komplexita tohto produktu môže sťažovať jeho pochopenie a naučenie.

Spomedzi open source projektov poskytuje Mule najširšie množstvo adaptérov na existujúce systémy, najlepšie riešenú bezpečnosť a má relatívne dobrú dokumentáciu v porovnaní s ostatnými open source produktmi. OpenESB ponúka najlepšie nástroje na správu. ServiceMix sa drží skôr svojej domény - poskytnúť kvalitný odľahčený kontajner JBI a podobne ako Mule, aj ServiceMix má dobré napojenie na iné produkty (MOM, aplikačné servery a pod.).

Vlastnosti analyzovaných produktov zhrnieme do prehľadnej tabuľky. Jednotlivé vlastnosti hodnotíme sumárne pomocou bodiek, slovne alebo kombináciou týchto dvoch metód. Pri hodnotení bodkami, reprezentuje jedna bodka najnižšie a päť bodiek najväčšie hodnotenie v danej vlastnosti. Radi by sme poznamenali, že v rámci

možností tejto práce je hodnotenie bodkami subjektívne. Pri hodnotení sa zaujímame o vlastnosti:

- **Začiatok vývoja**: mesiac a rok, v ktorom začal byť produkt vyvíjaný.
- **Dokumentácia**: dostupnosť, kvalita a konzistentnosť dokumentácie k produktu.
- **Komerčná podpora a tréning**: je k produktu dostupná komerčná podpora, či tréning?
- **Konektivita**: koľko existujúcich adaptérov, či konektorov produkt poskytuje, ich kvalita.
- **Bezpečnosť**: aká je úroveň zabezpečenia, ktorú produkt podporuje?
- **Robustnosť**: ako dobre rieši produkt robustnosť, dostupnosť, odozvu a súčasné prístupy pomocou škálovateľnosti, vyvažovania záťaže na služby, či komunikačnú vrstvu, ponúka *failover* pri výpadku servera?
- **Nástroje na vývoj, nasadzovanie a správu**: koľko a akých nástrojov na vývoj, nasadzovanie a správu produkt ponúka a koľko funkcionality poskytujú.
- **Integrácia s inými produktmi**: ako dobre sa vie produkt prepájať s inými produktmi (MOM, aplikačné servery a pod.)
- **Choreografia**: podporuje produkt choreografiu, tj. smerovanie na základe itinerára?
- **Orchestrácia**: podporuje produkt vlastné orchestrované procesy?
- **Ľahká naučiteľnosť**: ako ľahké a priamočiare je porozumieť produktu a naučiť sa ho používať.
- **Licencia**: pod akou licenciou je produkt poskytovaný.

	Sonic ESB	Apache ServiceMix	Open ESB	Mule
Začiatok vývoja	Marec 2002	August 2005	Máj 2006	December 2004
Dokumentácia	●●●●	●●	●	●●●
Komerčná podpora a tréning (dostupnosť)	áno	áno	nie	áno
Konektivita	●●●●●	●●	●●●	●●●●
Bezpečnosť	●●●●	●●	●●	●●●●
Robustnosť	●●●● (*clustering*)	●●● (*clustering*)	●●● (*clustering*)	●●● (SEDA, chýba podpora pre *failover*)
Nástroje na vývoj, nasadzovanie a správu	●●●●● (Eclipse, vlastné)	●● (Eclipse, Ant, Maven, Spring)	●●● (NetBeans)	●● (Eclipse, Ant, Maven, Spring, vlastné)
Integrácia s inými produktmi	●●●	●●●●	●●	●●●●
Choreografia	Áno	Nie	Nie	Áno
Orchestrácia	Áno, vlastný formát	Áno, BPEL	Áno, BPEL	Nie, len integrácia s externým systémom
Ľahká naučiteľnosť	●	●●	●●	●●●
Licencia	Komerčná	Apache 2.0	CDDL	MPL+

Tabuľka 2 - Zhrnutie vlastností analyzovaných produktov

Záver

Zhrnutie

Cieľom tejto práce bolo vykonať prieskum v oblasti ESB. V tejto práci prinášame ucelený pohľad na problematiku ESB, vysvetľujeme problematické miesta, prinášame definíciu ESB a zoznam hlavných čŕt. Popisujeme špecifikáciu Java Business Integration a jej význam pre oblasť ESB. Na modelovom príklade informačného systému Univerzity Komenského definujeme kritériá pre integrovaný informačný systém. V práci analyzujeme štyri produkty typu ESB pomocou celkového pohľadu, pomocou troch úrovní identifikovaných pri definovaní architektúry ESB a pomocou spôsobu, akým realizujú požiadavky na integrovaný informačný systém UK.

Použitie ESB v informačnom systéme UK

Pri vhodnom použití technológie ESB na integráciu informačného systému UK by sa mohli prejaviť výhody architektúry ESB zosumarizované v tejto práci. Integrácia by sa dala založiť na postupnom budovaní zbernice a postupnom pripájaní služieb k zbernici. Služby by sa mohli tvoriť iteratívnym spôsobom, nakoľko by boli voľne prepojené a nová verzia služby by nevyžadovala meniť ostatné služby pripojené na zbernicu. Pri pripájaní existujúcich systémov vo forme služieb na zbernicu by sa mohla využiť práca vykonaná v oblasti komunikácie pomocou správ JMS. Komplexné procesy prebiehajúce na univerzite by mohli byť v maximálnej možnej miere automatizované pomocou nástrojov podporujúcich procesné riadenie.

Použitie niektorého z produktov ESB by však pravdepodobne znamenalo nutnosť zaškolenia alebo vytrénovania pracovníkov zodpovedných za integráciu. Ako sme v práci spomenuli, niektoré produkty (napríklad Sonic ESB) majú pomerne strmú krivku naučiteľnosti (*learning curve*). Použitiu procesného riadenia by muselo predchádzať analyzovanie a definovanie súčasných procesov.

Medzi prvé kroky v integrácii pomocou ESB by mohla patriť napr. implementácia univerzálnych číselníkov, kde by sa vybudovala služba pracujúcu s univerzálnymi číselníkmi a táto služba by sa pripojila na komunikačnú zbernicu. Iné služby môžu byť pripojené na zbernicu neskôr a môžu využívať funkcionalitu služby univerzálnych číselníkov. Ďalším krokom by mohlo byť prepojenie niektorých existujúcich systémov pomocou ESB. Takýmto iteratívnym spôsobom sa dá postupne zaviezť ESB do prostredia UK a na základe toho vybudovať integrovaný informačný systém spĺňajúci všetky požiadavky.

Možnosti ďalšieho rozpracovania práce

Nakoľko architektúra ESB nie je presne definovaná, na niektoré produkty sa úrovne, na ktorých sme ich porovnávali, hodili viac ako na iné. Napríklad úrovne poskytované produktmi Sonic MQ, Sonic ESB a Sonic Orchestration Server sa pomerne presne zhodovali s tromi úrovňami architektúry ESB. Naproti tomu Mule, ktorý ako jediný nepoužíval nejaký konkrétny produkt typu MOM a neposkytoval priamo funkcionalitu procesnej orchestrácie sa na tieto úrovne mapoval menej priamočiaro. Pre prípadné ďalšie rozpracovanie by mohlo byť zaujímavé nezaložiť porovnanie na úrovniach, ale skôr na konkrétnych črtách daných produktov. Takisto by bolo možné

ísť pri porovnávaní produktov viac do hĺbky a spraviť napríklad prehľad konkrétnych spôsobov implementácie daných čŕt (kontajnerov, koncových bodov a podobne). Pri takejto analýze by sa dala naplno využiť dostupnosť zdrojového kódu open source produktov. Taktiež by bolo možné otestovať ostatné produkty typu ESB, ktoré sú dostupné na trhu. Väčšina komerčných produktov ponúka možnosť využiť na obmedzený čas testovaciu verziu.

Ďalšou z možností výskumu by mohla byť analýza štandardov pre webové služby (označovaných súhrnne ako WS-*) a ich vzťahu k ESB.

Slovník pojmov

Apache Geronimo – Open source aplikačný server J2EE. http://geronimo.apache.org

Apache Maven – Open source nástroj na centrálnu správu projektov - vytváranie (*build*), reportovanie, dokumentácia, nasadzovanie, testovanie. http://maven.apache.org/

API – Application Programming Interface – rozhranie na úrovni kódu, ktoré poskytuje systém, komponent a podobne.

B2B – Viď *business-to-business*.

BC – Binding Component – prepájací komponent. Prvok architektúry JBI. Viď 3.7 Java Business Integration.

BEA Weblogic – Komerčný aplikačný server J2EE. http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/

BPEL – Viď WS-BPEL.

BPM – Business Process Management – Oblasť zaoberajúca sa metódami, technikami a nástrojmi na dizajn, vytváranie, ovládanie a analyzovanie podnikových procesov.

broker – sprostredkovateľ. Viď architektúra *hub-and-spoke*.

business-to-business – Označenie komunikácie medzi dvoma alebo viacerými podnikmi.

business-to-business integration **–** Externá forma integrácie, pri ktorej sa prepájajú systémy dvoch alebo viacerých podnikov.

cluster – Skupina počítačov alebo procesov, ktoré spolupracujú takým spôsobom, že sa na ne dá pozerať ako na jednu entitu.

clustering – Viď *cluster*.

CBR – Content-Based Routing - Smerovanie správ podľa obsahu. Keď uzol v sieti dostane správu, smeruje ju ďalej na základe obsahu správy. Nevyžaduje od uzla znalosť nejakej centrálnej smerovacej politiky.

EAI – Enterprise Application Integration. Potreba prepojenia rôznych aplikácií a systémov v rámci podniku aby dosiahli cieľ jednotným a celistvým spôsobom, nezávisle na platforme a geografickej polohe jednotlivých aplikácií. Zahŕňa rôzne metodiky, vzory, architektúry, topológie a pod.

Eclipse – Open source platforma poskytujúca vývojárske prostredie pre mnohé programovacie jazyky, napríklad C/C++ alebo Java. http://www.eclipse.org/

EJB – Enterprise Java Bean. Spravovaný komponent pracujúci na strane servera, určený na modulárnu konštrukciu podnikových aplikácií v jazyku Java. http://java.sun.com/products/ejb/

endpoint – Viď koncový bod.

ESB – Enterprise Service Bus - Podniková zbernica služieb. Zbernicová architektúra, ktorá umožňuje realizáciu SOA princípov.

ETL – *Extract, Transfer, and Load* – Spôsob prenosu informácií medzi aplikáciami, pri ktorom sa informácie vytiahnu (*extract*) zo zdrojovej aplikácie a prenesú sa (*transfer*) k cieľovej aplikácii, kde sa nahrajú (*load*).

failover – Schopnosť počítačového systému automaticky sa prepnúť na záložný systém v prípade výskytu chyby.

framework – Opakovane použiteľný dizajn pre počítačový systém. Pozostáva z niekoľkých tried, zaobstarávajúcich špecifickú funkcionalitu (napríklad prístup k databáze), knižníc, podporných programov a podobne.

frontend – Časť softvérového systému, s ktorou prichádza do styku používateľ.

Geronimo – viď Apache Geronimo.

Glassfish – Open source verzia aplikačného servera J2EE Sun Java Systems Application Server (SJSAS) určená na vývoj novej funkcionality do SJSAS. https://glassfish.dev.java.net/

hub-and-spoke (architektúra) – Architektúra využívajúca centralizovaného sprostredkovateľa (*broker* alebo *hub*), na ktorý sa pripájajú aplikácie pomocou adaptérov (*spokes*). Federovaná *hub-and-spoke* architektúra využíva viacero sprostredkovateľov, smerovacie a iné pravidlá sa zdieľajú medzi sprostredkovateľmi.

choreografia - Druh procesného toku bez centrálneho riadenia. Využíva itinerár. Viď 3.4.3 Úroveň procesov.

IBM WebSphere - Komerčný aplikačný server J2EE. http://www-306.ibm.com/software/websphere/

itinerár - Postupnosť diskrétnych smerovacích operácií prenášaných so správou. Viď tiež choreografia.

IIKS – Integrovaný informačný a komunikačný systém (Univerzity Komenského) - Označenie pre súčasný informačný systém Univerzity Komenského.

IIS UK – Integrovaný informačný systém Univerzity Komenského – Označenie pre (budúci) integrovaný informačný systém Univerzity Komenského používané v tejto práci. Pre odlíšenie od IIKS viď kapitolu 2 Integrácia v informačnom systéme Univerzity Komenského.

IS UK – Informačný systém Univerzity Komenského.

J2EE – Java Platform, Enterprise Edition – Programová platforma Java, určená predovšetkým na vývoj a beh distribuovaných viac vrstvových podnikových aplikácií. http://java.sun.com/javaee/

J2SE – Java Platform, Standard Edition – Programová platforma Java, určená predovšetkým na vývoj a beh *desktop*, či vsadených (*embedded*) aplikácií. http://java.sun.com/javase/

JAAS – Java Authentication and Authorization Service – Bezpečnostný *framework* na autentifikáciu a autorizáciu používateľov v jazyku Java. http://java.sun.com/products/jaas/

JavaBean - komponent v programovacom jazyku Java, tvorený niekoľkými triedami, ktoré spĺňajú istú konvenciu. http://java.sun.com/products/javabeans/

JBI – Java Business Integration – Špecifikácia architektúry. Viď 3.7 Java Business Integration.

JBoss - Viď Red Hat JBoss.

JDK – Java Development Kit – Sada nástrojov na vývoj a spúšťanie programov v jazyku Java od spoločnosti Sun. Medzi tieto nástroje patrí kompilátor, JVM, nástroj na tvorbu dokumentácie a pod. Základná verzia je súčasťou programovej platformy J2SE.

JMS – Java Message Service – Štandardné API pre produkty typu MOM. http://java.sun.com/products/jms/

JMX – Java Management eXtensions – Java technológia určená na správu a monitorovanie objektov a aplikácií. http://java.sun.com/products/JavaManagement/

JNDI – Java Naming and Directory Interface – Štandardné API pre adresárové služby, ktoré umožňuje vyhľadávanie objektov podľa ich názvu. http://java.sun.com/products/jndi/

JVM – Java Virtual Machine – Virtuálny stroj, ktorý interpretuje programy v programovacom jazyku Java.

koncový bod – Rozhranie, pomocou ktorého komunikuje služba s ostanými službami. Poskytuje abstrakciu nad použitým komunikačným protokolom, umiestnením služby a pod.

LDAP – Lightweight Directory Access Protocol – Aplikačný protokol určený na prácu s adresárovými službami postavený nad TCP/IP.

load balancer – Zariadenie zaobstarávajúce vyvažovanie záťaže (*load balancing*).

load balancing – vyvažovanie záťaže – Spôsob rozdeľovania záťaže medzi viacero identických komponentov, systémov a podobne, spravidla pracujúcich v *clustri*.

Maven - viď Apache Maven

MEP – Message Exchange Pattern – vzor výmeny správ – Spôsob komunikácie medzi dvoma službami použitý v JBI. Viď 3.7.1 Úroveň správ.

middleware – Systém, ktorý poskytuje medzivrstvu, pomocou ktorej sa prepájajú iné systémy aby mohli zdieľať informácie.

MOM – Message-Oriented Middleware – *Middleware* založený na výmene správ. Poskytuje funkcionalitu posielania a prijímania správ v rôznych módoch. Viď 3.5.2 MOM - Message Oriented Middleware.

NetBeans - Open source platforma poskytujúca vývojárske prostredie pre jazyk Java. http://www.netbeans.org/

NMR – Normalized Message Router – smerovač normalizovaných správ. Prvok architektúry JBI. Viď 3.7 Java Business Integration.

Orchestrácia - Druh procesného toku s centrálnym riadením. Viď 3.4.3 Úroveň procesov.

peer-to-peer – Označenie priameho prenosu medzi dvoma systémami alebo službami, bez použitia sprostredkovateľa.

PKI – Public Key Infrastructure – Kombinácia softvéru, šifrovacích technológií a služieb, ktorá rieši otázky v oblasti bezpečnosti s využitím asymetrickej kryptografie.

plugin architecture – zapojiteľná architektúra – Architektúra, v ktorej sú jednotlivé komponenty voľne prepojené a môžu byť zapájané postupne.

POJO - *Plain Old Java Object* – Objekt v programovacom jazyku Java, od ktorého sa nevyžaduje spĺňanie príliš striktných a komplikovaných pravidiel (ako sú napríklad pravidlá pre triedy EJB).

Red Hat JBoss - Open source aplikačný server J2EE. http://www.jboss.org

SE – Service Engine – službový prostriedok. Prvok architektúry JBI. Viď 3.7 Java Business Integration.

SEDA - *Staged Event-Driven Architecture* – Dizajnový vzor na vytváranie vysoko výkonných a škálovateľných distribuovaných systémov. http://www.eecs.harvard.edu/~mdw/proj/seda/

single point of failure – bod zlyhania – Systém má jedno slabé miesto, pri ktorého zlyhaní zlyhá celý systém.

single sign-on – Forma autentifikácie, pri ktorej sa používateľ nemusí autentifikovať voči každému systému zvlášť, ale autentifikuje sa iba jeden krát a systémy si informáciu o autentifikácii zdieľajú.

SJSAS - viď Sun Java System Application Server.

SOA – Service Oriented Architecture – Architektúra orientovaná na služby. Aplikácie, alebo funkcionality aplikácií sú sprístupnené ako služby na sieti, ktoré sa dajú vyhľadať a volať. Viac napr. v [KRA05].

SOAP – Protokol na výmenu správ založený na formáte XML. Ako transportná vrstva sa môže využívať napríklad protokol HTTP.

Spring Framework - Aplikačný *framework* pre jazyk Java, ktorý zahŕňa širokú paletu funkcionalít, od vytvárania objektov, cez rôzne kontajnery, až po správu konfigurácií a testovanie. Využíva a prepája mnohé iné existujúce *frameworky*, ktoré sa starajú napríklad o mapovanie objektov na relačné databázy, správu databázových pripojení a podobne. http://www.springframework.org/

SSL – Secure Socket Layer – Šifrovací protokol zabezpečujúci prenos pracujúci medzi transportnou vrstvou (napr. TCP) a aplikačnou vrstvou (HTTP).

stub – Čiastočná implementácia triedy, metódy a pod. Navonok má rovnaké charakteristiky ako skutočná implementácia (napr. signatúry metód).

Sun Java System Application Server - Komerčný aplikačný server J2EE. http://www.sun.com/software/products/appsrvr/index.xml

TLS – Transport Layer Security – Novšia verzia protokolu SSL.

UDDI – Universal Description Discovery and Integration – Adresár služieb založený na formáte XML.

UK – Univerzita Komenského v Bratislave.

Web Services – Viď webové služby.

Weblogic - viď BEA Weblogic.

webové služby – Architektúra umožňujúca definovanie rozhraní služieb, ich sprístupnenie cez sieť. Využíva protokoly XML, SOAP, WSDL, UDDI, WS-Reliability, WS-Notification a mnohé ďalšie.

WebSphere - Viď IBM WebSphere.

WS-* - Súborné označenie pre všetky špecifikácie webových služieb, ktoré začínajú „WS-“.

WS-BPEL - Web Service Business Process Execution Language od spoločnosti OASIS. Jazyk na popísanie procesov ako prepojených aktivít, ktoré sú

definované ako webové služby. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

WS-Notification – Špecifikácia umožňujúca udalosťami riadené (*event driven*) programovanie medzi webovými službami. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn

WS-Reliability – Špecifikácia zaoberajúca sa zaistením spoľahlivého prenosu informácií medzi webovými službami pomocou protokolu SOAP. Predchodca špecifikácie WS-ReliabilityMessaging.

WS-ReliableMessaging - Špecifikácia zaoberajúca sa bezpečnosťou prenosov informácií medzi webovými službami pomocou protokolu SOAP. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm

WS-Rel* - Súborné označenie pre špecifikácie WS-Reliability a WS-ReliableMessaging.

WSDL – Web Services Description Language – Jazyk založený na formáte XML pôvodne určený na popis rozhraní webových služieb. Verzia 1.1: http://www.w3.org/TR/wsdl, verzia 2.0: http://www.w3.org/TR/wsdl20/

X.509 – Bezpečnostný protokol pre PKI (Public Key Infrastructure). Okrem iného špecifikuje napríklad formát pre certifikáty.

XML – eXtensible Markup Language – Dátový formát použiteľný na široké spektrum aplikácií. http://www.w3.org/XML/

XPath – Jazyk na vyjadrovanie výrazov, ktoré vyberajú dáta z dokumentu XML. http://www.w3.org/TR/xpath

XQuery – Dotazovací jazyk pre XML. http://www.w3.org/XML/Query

XSLT – eXtensible Stylesheet Language Transformation – Jazyk založený na formáte XML, určený na transformáciu dokumentov XML. http://www.w3.org/TR/xslt20/

XSLT *stylesheet* – „Program“ napísaný v jazyku XSLT.

zbernicová architektúra – Architektúra využívajúca spoločnú zbernicu, na ktorú sa pripájajú aplikácie, resp. systémy.

Zoznam bibliografických odkazov

[BAK05] BAKKER, L. *Goodbye Hub-and-Spoke, Hello ESB? Integration Architecture With BizTalk 2004* [online]. September 2005. Článok na .NET Developer's Journal. Formát HTML. Dostupné na internete: <http://dotnet.sys-con.com/read/121831.htm>

[BIS06] BISTÁK, Ľ. *Technológie pre webové služby*. Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2006. Diplomová práca.

[CAPMES] *Service-Centric vs. Message-Centric ESBs* [online]. CapeClear. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.capeclear.com/technology/messaging.shtml>

[CAPTECH] *Understanding SOA, ESB and Web Services* [online]. CapeClear. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.capeclear.com/technology>

[CHA04] CHAPPELL, D. *Enterprise Service Bus*. O'Reilly, 2004. ISBN 0-596-00675-6.

[CHA05JBI] CHAPPELL, D. *Enterprise Service Bus and Java™ Business Integration: Infrastructure for Enterprise SOA* [online]. JavaOneSM Conference, Session TS-1428, 2005. Formát PDF. Dostupné na internete: <http://developers.sun.com/learning/javaoneonline/2005/coreenterprise/TS-1428.pdf>

[FRY06] FRYE, C. *ESB market report* [online]. SearchWebServices.com, Október 2006. Formát HTML. Dostupné na internete: <http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1226497,00.html>

[GAR03] Hy*pe Cycle for Application Integration and Platform Middleware, 2003* [online]. Formát HTML. Dostupné na internete: <http://www.gartner.com/DisplayDocument?id=396290>

[GAR04] *Hype Cycle for Application Integration and Platform Middleware, 2004* [online]. Formát HTML. Dostupné na internete: <http://www.gartner.com/DisplayDocument?doc_cd=120915>

[GAR05] *Hype Cycle for Application Integration and Platform Middleware, 2005* [online]. Formát HTML. Dostupné na internete: <http://www.gartner.com/DisplayDocument?doc_cd=127756>

[GAR06] *Hype Cycle for Application Integration and Platform Middleware, 2006* [online]. Formát HTML. Dostupné na internete: <http://www.gartner.com/DisplayDocument?id=493861>

[GOEL] GOEL A. *Enterprise Integration: EAI vs. SOA vs. ESB* [online]. Whitepaper. Formát PDF. Dostupné na internete:

<http://hosteddocs.ittoolbox.com/Enterprise Integration - SOA vs EAI vs ESB.pdf>

[GOP06] GOPALAN, S. R., BINROD, P.G., BABO, K., PALKOVIC, R. *Implementing Service-Oriented Architectures (SOA) with the Java EE 5 SDK* [online]. Článok na Sun Developer Network, Máj 2006. Formát PDF. Dostupné na internete: <http://java.sun.com/developer/technicalArticles/WebServices/soa3/ImplementingSOA.pdf>

[HAR06] HARBY, J. *ESB Alternative* [online]. November 2006. Článok na InfoQ. Formát HTML. Dostupné na internete: <http://www.infoq.com/articles/ESB-alternative;jsessionid=A8C2F087720E4317F0A873FE7961A3DC>

[HOH04] HOHPE, G., WOOLF, B. *Enterprise Integration Patterns.* Addison-Wesley, 2004. ISBN 0-321-20068-3

[INT07] *Konkretizácia obsahu, rozsahu a zodpovednosti v rozvojovej/prevádzkovej úlohe IIKS v roku 2007*, kódy INT1 až INT4. Interný materiál Centra informačných technológií Univerzity Komenského, 2007.

[INT06] *Konkretizácia obsahu, rozsahu a zodpovednosti v rozvojovej/prevádzkovej úlohe IIKS v roku 2006*, kód INT2. Interný materiál Centra informačných technológií Univerzity Komenského, 2006.

[JBIFOR] *Developer Forums: Enterprise Technologies - Java Business Integration (JBI)* [online]. Vývojárske fórum. Formát HTML. Dostupné na internete: <http://forum.java.sun.com/forum.jspa?forumID=512> [cit. 10. apríla 2007]

[JBIWIK] *Community Wiki for Java Business Integration (JBI), JBI Component Development and Open ESB* [online]. Komunitné wiki. Formát HTML. Dostupné na internete: <http://www.glassfishwiki.org/jbiwiki/> [cit. 10. apríla 2007]

[JSRJBI] *JSR 208: Java Business Integration (JBI)* [online]. Návrh na špecifikáciu štandardu. Formát HTML. Dostupné na internete: <http://www.jcp.org/en/jsr/detail?id=208>

[KOP07] KOPÁČ, P. *Jednotná autentifikácia používateľov webových aplikácií na UK*. Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2007. Diplomová práca. V tlači.

[KRA05] KRAFZIG, D., BANKE, K., SLAMA, D. *Enterprise SOA.* Prentice Hall, 2005. ISBN 0-131-46575-9

[MAR06] MARKS, E., BELL M. *Service-Oriented Architecture.* New Jersey : John Wiley & Sons, Inc., 2006. ISBN 0-471-76894-4

[MEDAR07] MEDERLY, P. *Architektúra IIKS*. Interný materiál Centra informačných technológií Univerzity Komenského, 2007.

[MEDIA06] MEDERLY, P. *Integrácia aplikácií na Univerzite Komenského*. In Rozvoj informačných technológií na slovenských vysokých školách - RIKT 2006. Nitra : Centrum informačných technológií FEM SPU v Nitre, 2006. ISBN 80-8069-804-X.

[MULE] *Mule* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://mule.codehaus.org/>

[OPEN] *Project Open ESB* [online]. Komunitná domovská stránka produktu. Formát HTML. Dostupné na internete: <https://open-esb.dev.java.net/>

[OPEN2B] *Open ESB 2.0 Beta* [online]. Domovská stránka produktu na Sun Developer Network. Formát HTML. Dostupné na internete: <http://java.sun.com/integration/openesb2_0/>

[PÁL06] PÁLOS, G. *Komunikácia aplikácií v informačnom systéme Univerzity Komenského.* Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2006. Diplomová práca.

[SMIX] *Apache ServiceMix* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.servicemix.org>

[SMIXJBI] *Apache Service Mix, JBI* [online]. Dokumentácia k produktu. Formát HTML. Dostupné na internete: <http://incubator.apache.org/servicemix/5-jbi.html>

[SVA07] SVAČINA, O. *Proces integrácie aplikácií* (predbežný názov). Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2007. Diplomová práca. V tlači.

[SONDOC] *Sonic ESB Product Documentation* [online]. Dokumentácia k produktu. Formát PDF. Dostupné na internete: <http://www.sonicsoftware.com/products/sonic_esb/documentation/index.ssp>

[SONDEV] *Sonic ESB Product Family Developer's Guide*. Interná dokumentácia k produktu Sonic ESB Product Family, nainštalované ako súčasť Sonic ESB 7.0. Formát HTML.

[SJSMQ] *Sun Java System Message Queue 3 2005Q1 Technical Overview* [online]. Dokumentácia k produktu. Formát HTML. Dostupné na internete: <http://docs.sun.com/app/docs/doc/819-0069>

[TER06] TERKANIČ, J. *Zdieľanie informácií o osobách v informačnom systéme Univerzity Komenského.* Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2006. Diplomová práca.

[VDO06] VAN DONGE, T. *Building an Agile Enterprise*. [online]. Cordys, Január 2006. Whitepaper. Formát PDF. Dostupné na internete: <http://www.cordys.com/ufc/file2/cordyscms_sites/tarun/93a7d3396ba997b33e3a8c8fb6011d00/pu/Building_an_Agile_Enterprise.pdf>

[VOZ05] VOZÁR, M., PETRÍK, J., MEDERLY, P. *Študent NG*. Špecifikácia softvéru, Verzia 1.0. Interný materiál Centra informačných technológií Univerzity Komenského, November 2005.

Prílohy

A. Prehľad komerčných a open source produktov typu ESB, október 2006

Prevzaté z [FRY06].

Spoločnosť	**Iona Technologies**, Waltham, Mass., USA, www.iona.com	**Sonic Software Corp.**, spoločnosť vlastnená Progress Software Corp., Bedford, Mass., USA, www.sonicsoftware.com	**BEA Systems Inc.**, San Jose, Calif., USA, www.bea.com	**Cape Clear Software Inc.**, Waltham, Mass., USA, www.capeclear.com	**Fiorano Software Inc.**, Los Gatos, Calif., USA, www.fiorano.com	**Oracle Corp.**, Redwood Shores, Calif., USA, www.oracle.com
Spoločnosť založená	1991	2001	1995	1999	1995	1977
Názov produktu	**Artix**	**Sonic ESB**	**BEA AquaLogic Service Bus**	**Cape Clear ESB**	**Fiorano Enterprise Service Bus**	**Oracle Enterprise Service Bus**
Prvý štart	Október 2003	Marec 2002	Jún 2005	December 2000	Apríl 2003 (Tifosi 2002 ESB)	3. kvartál 2003
História	Korene v integrácii, pôvodne CORBA pomocou ich produktovej línie Orbix, neskôr webové služby pomocou Artix ESB a sponzorovanie open source Celtix ESB	Boli im pripísané zásluhy za prvé ESB (pôvodný názov SonicXQ) a za zavedenie produktovej kategórie ESB. Spoločnosť odkúpená spoločnosťou Progress Software.	Korene v trhu s aplikačnými servermi a middleware s produktom WebLogic, rozšírili na trh platforiem SOA uvedením produktovej línie AquaLogic.	Korene v integrácii, prvý produkt spoločnosti bol XML Business Server pre sprístupnenie J2EE a CORBA komponentov ako webové služby.	Korene ako serverovo založená správová middleware platforma postavená na Java Message Service (JMS). Fiorano ESB je základ ich platformy SOA.	Korene v relačných databázach a middleware, Oracle rozšíril svoje aplikácie o SOA. Oracle ESB je súčasť Oracle Application Server Integration Platform, kľúčového komponentu Oracle Fusion Middleware.
Cenová relácia	Cena za Artix prostredie je 10.000 USD za CPU, ďalšie Artix *pluginy* sa pohybujú medzi 2.500 USD za CPU do 10.000 USD za CPO	začína od 10.000 USD	20.000 až 30.000 USD za CPU	75.000 USD a viac	ESB Server - 25.000 USD, platforma SOA 2006 (obsahuje ESB Server) - 40.000 USD	20.000 až 50.000 USD za CPU (momentálne ESB je predávané samostatne za 20.000 USD za CPU alebo ako súčasť Oracle SOA Suite za 50.000 USD za CPU)

Tabuľka 3 - Komerčné ESB, časť 1.

Spoločnosť	**IBM Corp.**, Armonk, N.Y., USA, www.ibm.com	**Sun Microsystems Inc.**, Santa Clara, Calif., USA, www.sun.com	**Tibco Software Inc.**, Palo Alto, Calif., USA, www.tibco.com	**WebMethods Inc.**, Fairfax, Va., USA, www.webmethods.com	**PolarLake Ltd.**, Dublin, Írsko, www.polarlake.com	**Software AG**, Darmstadt, Nemecko a Reston, Va., USA, www.softwareag.com	**Cordys**, Chicago, USA a Holandsko, www.cordys.com
Spoločnosť založená	1911	1982	1985 (ako Teknekron Software Systems)	1996	1999	1969	2001
Názov produktu	**IBM WebSphere Enterprise Service Bus**	**Java Enterprise Service Bus Suite** (časť Java Composite Application Platform Suite)	**Tibco BusinessWorks**	**webMethods Fabric Enterprise Services Platform**	**PolarLake Messaging Integrator**, časť PolarLake Integration Suite	**crossvision Service Orchestrator**	**Cordys Enterprise Service Bus** (časť Application Platform Suite)
Prvý štart	September 2005	Marec 2006	December 2001	Október 2004	Jún 2003	1999	2001
História	Približovala sa trhu SOA od počiatkov správových middleware produktov, rozširujúc svoju platformu aplikačných serverov WebSphere. IBM taktiež ponúka služby a konzultovanie ohľadom SOA.	Java ESB Suite obsahuje integračné produkty a technológie od akvizície SeeBeyond spolu s ostatnými produktmi Java Enterprise System.	Korene v integrácii EAI a B2B, dnes sa upriamujú na SOA, *business process managemeng* (BPM) a podnikovú optimalizáciu.	Korene v integrácii EAI a B2B, spojili sa s Active Software v roku 2000, rozšírili sa na trh BPM, v septembri 2006 oznámili plány na kúpenie dodávateľa registru/skladu Infravio.	Korene v integrácii pomocou XML, vydali PolarLake verziu 1.0, podnikovú platformu XML pre Javu v septembri 2001.	Korene v DBMS s Adabas-om, vyvinuli Natural, 4GL prostredie na vývoj aplikácií, pridali integráciu a platformy XML, partneri s Fujitsu a IDS Scheer na dodanie platformy SOA.	Založené Janom Baanom, ktorý tiež založil ERP softvérovú spoločnosť Baan Co. a neskôr TopTier, ktorá bola kúpená SAPom. Technológia Cordys je založená na portálovej a integračnej technológii TopTier.
Cenová relácia	25.000 USD (vrátanie 12 mesačnej údržby)	50 USD za zamestnanca a rok pomocou predplatného	Začiatočná cena 75.000 USD za CPU	Typická cena začína od 100.000 USD.	Spoločnosť neodpovedala na požiadavky o informácie.	50.000 USD	100 USD za používateľa (predplatné) alebo 100.000 USD za CPU (cena za Suite, ESB samostatne nepredajné)

Tabuľka 4 - Komerčné ESB, časť 2.

Názov produktu	Celtix	Mule	Apache ServiceMix	Open ESB	JBoss ESB
Komunita/Sponzor	ObjectWeb/Iona	MuleSource	Apache Foundation	Sun Microsystems Inc.	Red Hat (JBoss)
História	Projekt sponzorovala spoločnosť Iona Technologies a dodala aj technológiu Artix ESB. Celix podporuje viacero kontajnerov (J2EE, servlet, samostatný), taktiež podporuje ActiveMQ Java Message Service (JMS).	Myslené ako odľahčený správový framework. J2EE 1.4 ESB a správový sprostredkovateľ, prepojiteľné s JMS a JBI. Pôvodne sponzorované spoločnosťou Symphony Soft (teraz pod názvom Ricston), teraz podporované MuleSource.	Od základov postavené na JBI špecifikácii, vydané pod licenciou Apache. Spočiatku podporované spoločnosťou LogicBlaze, ktorá na ServiceMix-e postavila svoju platformu Fuse SOA. Spustené ako inkubačný projekt ASF v decembri 2005.	Založené na JBI špecifikácii. Engine JBI a niekoľko komponentov z projektu OpenESB sú obsiahnuté v Java EE 5 SDK, ktorý bol spustený v roku 2006 na JavaOne a asociované nástroje boli vydané v betaverzii ako súčasť NetBeans 5.5 a Enterprise Pack, tiež na JavaOne.	V októbri 2005 oznámil JBoss svoju stratégiu SOA a položil svoju JBoss Enterprise Middleware Suite (JEMS) ako open source platformu pre SOA. Je plánované rozšírenie JEMS aby obsahovalo ESB.
Prvý štart	Máj 2006	December 2004	August 2005	Skorý prístup máj 2006	Betaverzia august 2006

Tabuľka 5 - Open source ESB

B. Rozhrania systémov v rámci IS UK

Vypracované s pomocou pracovníkov Centra informačných technológií UK, marec - apríl 2007. Táto príloha reprezentuje jednu tabuľku, rozdelenú na štyri časti kvôli dostupnému miestu. Tabuľka ponúka rámcový prehľad o systémoch, platformách na ktorých systémy bežia a rôznych spôsoboch komunikácie, ktoré systémy podporujú. K termínu odovzdania tejto práce sa nepodarilo zistiť informácie o niektorých rozhraniach, tabuľka preto nie je finálna. Tabuľka teda slúži na ilustráciu niektorých dostupných možností komunikácie so systémami.

Detailnejší popis niektorých stĺpcov (ostatné stĺpce majú samovysvetľujúce názvy):

Stĺpec	**Popis**
Databáza – Vieme čítať	Vieme prečítať údaje z databázy aplikácie alebo systému a zachovať ich sémantickú hodnotu, akú mali v systéme?
Databáza – Vieme písať, aby to aplikácia prečítala	Vieme zapísať údaje do databázy aplikácie alebo systému tak, aby s nimi dokázala aplikácia alebo systém pracovať?
Lokálny súborový systém – Aplikácia vie exportovať	Vie aplikácia exportovať svoje údaje vo forme súborov na lokálny súborový systém?
Lokálny súborový systém – Aplikácia vie importovať (niektoré údaje)	Vie aplikácia importovať aspoň niektoré údaje zo súborov umiestnených na lokálnom súborovom systéme?
HTTP – Aplikácia vie zobraziť údaje ako parsovateľné (X)HTML	Vie aplikácia zobraziť svoje údaje vo forme HTML alebo XHTML stránky, z ktorej sa dajú tieto údaje automatizovane spracovať a načítať, napríklad pomocou parsovania?

Tabuľka 6 - Popis niektorých stĺpcov v prílohe B

Ak je v políčku hodnota „konfig.“, znamená to, že aplikácia sa dá prispôsobiť pomocou konfigurácie, netreba meniť jej kód.

Aplikácia / Systém	Krátky popis	Platforma	Je aplikácia online - prístupná cez sieť?	Je aplikácia modifikovateľná?	Podporované rozhrania			
					Databáza		Lokálny súborový systém	
					Vieme čítať	Vieme písať, aby to apl. prečítala	Apl. vie exportovať	Apl. vie importovať (niektoré údaje)
FIS	Finančný informačný systém	SAP R/3	áno	konfig.	nie	nie	ručný export	ručne
Študent	Študijná agenda	Clipper/DBF	nie	áno	ťažko	ťažko	áno (ručne/automaticky)	áno (ručne/automaticky)
Študent ADM	Sprístupnenie údajov o študijnej agende zo systému Študent (vo vývoji)	MS SQL	áno	áno	áno	áno	nie	nie
Študent II	Študijná agenda (plánované)	v súčasnosti neznáma, prebieha verejné obstarávanie	áno	čiastočne	áno	možno	ručný export	možno
Anketa	Anketa	PHP	áno	čiastočne	nie	nie	nie	áno
VIKUK	Virtuálna knižnica	VIRTUA	áno	len málo	áno	nie	ručný export	ručný import
CDO	Centrálna databáza osôb a preukazov	Java, MS SQL	áno	áno	áno	teoreticky áno, preferujeme správy	áno	teoreticky áno, preferujeme správy
PCard	Tlač preukazov	Java	nie	áno	áno	áno	áno	áno

Tabuľka 7 - Rozhrania systémov v IS UK, časť 1.

Aplikácia / Systém	Krátky popis	Platforma	Je aplikácia online - prístupná cez sieť?	Je aplikácia modifikovateľná?	Podporované rozhrania			
					Databáza		Lokálny súborový systém	
					Vieme čítať	Vieme písať, aby to apl. prečítala	Apl. vie exportovať	Apl. vie importovať (niektoré údaje)
JAS	Jednotný autentifikačný systém (vo vývoji)	Kerberos/LDAP/Sun Access Manager	áno	nie	nie	nie	áno	áno
PS	Prístupový systém	Visual Basic, MS SQL	áno	nie	áno	nie	áno	áno
DS	Dochádzkový systém RUK	Visual Basic	áno	nie	áno	nie	áno	áno
UT	Univerzitné validačné terminály	špecializovaný hardvér a softvér	áno	nie	nie	nie	nie	nie
ELEA	E-learning (viac inštancií)	PHP, rôzne DB (moodle)	áno	len málo	?	?	?	?
HDSW	Helpdesk	J2EE, Postgres	áno	áno	áno	nie	čiastočne (možno ručný export)	áno
Granty	Zobrazenie informácií o vedeckých grantoch na webe	MySQL, PHP	áno	áno	áno	áno	nie	nie
VoIP	Účtovanie Voice over IP hovorov	PHP, MySQL	áno	áno	áno	áno	nie	nie

Tabuľka 8 - Rozhrania systémov v IS UK, časť 2.

Aplikácia / Systém	Podporované rozhrania (pokračovanie)							Poznámky
	FTP (import/export)		TCP/IP	HTTP	SOAP správy	JMS správy	Iné	
	Apl. vie exportovať na FTP server	Apl. vie importovať z FTP servera	S apl. sa dá nadviazať spojenie	Apl. vie zobraziť údaje ako parsovateľné (X)HTML	Apl. vie prijímať a posielať	Apl. vie prijímať a posielať		
FIS	nie	nie	nie	nie	áno, cez PI	áno, cez PI	Remote Function Call	PI = SAP NetWeaver Process Integration
Študent	nie	nie	nie	nie	nie	nie	-	
Študent ADM	nie	nie	nie	nie	nie	nie	-	v súčasnosti je to "pasívna" databáza bez aplikačnej logiky
Študent II	nie	nie	možno	pravdepodobne áno	možno	možno	-	možno – v závislosti od vybraného riešenia
Anketa	nie	nie	nie	nie	nie	funkcionalita je vo vývoji	momentálne posielanie údajov mailom	
VIKUK	nie	nie	nie	pravdepodobne áno (simulácia činnosti používateľa)	nie	áno (import dát o študentoch)	-	
CDO	nie	nie	nie	áno	nie	áno	-	
PCard	nie	nie	nie	nie	nie	je možné dodatočne implementovať	-	

Tabuľka 9 - Rozhrania systémov v IS UK, časť 3.

Aplikácia / Systém	Podporované rozhrania (pokračovanie)							Poznámky
	FTP (import/export)		TCP/IP	HTTP	SOAP správy	JMS správy	Iné	
	Apl. vie exportovať na FTP server	Apl. vie importovať z FTP servera	S apl. sa dá nadviazať spojenie	Apl. vie zobraziť údaje ako parsovateľné (X)HTML	Apl. vie prijímať a posielať	Apl. vie prijímať a posielať		
JAS	nie	nie	nie	nie	nie	nie	-	detaily sa ujasnia v priebehu implementácie
PS	nie	nie	nie	nie	nie	nie	-	
DS	nie	nie	nie	pravdepodobne áno (simulácia činnosti používateľa)	nie	nie	-	
UT	nie	nie	áno	nie	nie	nie	-	špeciálny formát (potrebný uploadovací program)
ELEA	?	?	?	?	?	?	?	? = v súčasnosti nevieme
HDSW	nie	nie	nie	pravdepodobne áno (simulácia činnosti používateľa)	nie	áno	-	
Granty	nie	nie	nie	pravdepodobne áno (simulácia činnosti používateľa)	nie	nie	-	aplikácia sa v blízkej dobe prestane používať
VoIP	nie	nie	nie	pravdepodobne áno (simulácia činnosti používateľa)	nie	nie	-	

Tabuľka 10 - Rozhrania systémov v IS UK, časť 4.