FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

Komunikácia aplikácií v informačnom systéme Univerzity Komenského

Diplomová práca

Gustáv Pálos **2006**

Komunikácia aplikácií v informačnom systéme Univerzity Komenského

DIPLOMOVÁ PRÁCA

Gustáv Pálos

UNIVERZITA KOMENSKÉHO
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Vedúci záverečnej práce
Mgr. Pavol Mederly

BRATISLAVA 2006

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

V Bratislave .....
máj 2006 Gustáv Pálos

Ďakujem svojmu diplomovému vedúcemu, Mgr. Pavlovi Mederlymu za cenné rady, pripomienky, podnety, poskytovanie potrebných zdrojov a vedenie pri písaní tejto práce.

Ďakujem Univerzite Komenského, správcovi systému Študent, Ing. Jánovi Petríkovi, správcovi centrálnej databázy osôb, Jánovi Terkaničovi a správcom fakultných sietí univerzity za spoluprácu počas nasadenia a prevádzky.

Ďakujem svojim rodičom, sestre, kamarátke a kamarátom za vytvorenie tých najlepších podmienok pre vznik práce a tiež všetkým, ktorí mi akýmkoľvek spôsobom pomohli.

Abstrakt

Predložená diplomová práca sa zaoberá technickými riešeniami pre výmenu informácií medzi subsystémami informačného systému, konkrétne ide o posielanie správ prostredníctvom Java Message Service (JMS). Hlavnými cieľmi boli výber vhodnej nekomerčnej implementácie JMS a vytvorenie adaptérov pre zaistenie komunikácie.

Práca čerpá inšpiráciu z viacerých porovnaní implementácií JMS, ktoré sa však z rôznych dôvodov ukázali byť pre účely práce nedostatočné.

V práci sú na základe reálnych požiadaviek Univerzity Komenského porovnané vlastnosti vybraných poskytovateľov JMS a následne sú niektoré vlastnosti najrelevantnejších produktov aj prakticky otestované. Práca obsahuje zároveň popis realizácie dvoch adaptérov, ktoré slúžia na sprostredkovanie komunikácie medzi príslušnými subsystémami informačného systému univerzity. Tieto adaptéry boli spolu s vybraným produktom uvedené do prevádzky a používané ku dňu odovzdania práce 8 mesiacov.

Hlavným prínosom tejto práce je porovnanie vlastností produktov JMS, otestovanie korektnosti ukladania perzistentných správ v týchto produktoch a vytvorenie spomínaných adaptérov.

Kľúčové slová

integrácia aplikácií, Java Message Service, porovnanie implementácií JMS, spoľahlivosť, adaptér

Predhovor

Cieľom práce je navrhnúť a čiastočne realizovať technické riešenie pre výmenu informácií medzi vybranými subsystémami informačného systému Univerzity Komenského. Úlohou bolo sformulovať popis problému vrátane konkrétnych požiadaviek na jeho riešenie, navrhnúť riešenie, následne vybrať, resp. vytvoriť potrebné komponenty a vytvorené dielo uviesť do testovacej prevádzky v prostredí informačného systému univerzity.

Hlavným prínosom tejto práce je prehľad charakteristík vybraných voľne dostupných implementácií štandardu Java Message Service (JMS), testovanie korektnosti ukladania perzistentných správ vo výnimočných situáciách, s ktorým sa môžeme stretnúť počas prevádzky a implementácia dvoch adaptérov umožňujúcich komunikáciu troch konkrétnych aplikácií prostredníctvom JMS. Táto práca môže slúžiť ako základ pre riešenie komunikácie medzi subsystémami informačného systému v rámci UK, resp. v iných organizáciách[1]. Výsledky práce sú zároveň významné aj v širšom kontexte, nakoľko počas písania diplomovej práce nebolo nájdené porovnanie podobného rozsahu, ani podobný test spoľahlivosti relevantných nekomerčných implementácií JMS.

S touto prácou úzko súvisí diplomová práca Jána Terkaniča [Ter06], ktorej súčasťou je aj návrh a čiastočná implementácia centrálnej databázy osôb (zaistenie komunikácie tejto databázy s ďalšími aplikáciami je predmetom tejto práce). Ďalšie príbuzné práce sú [Bis06], [Bal06].

Hlavnými informačnými zdrojmi použitými v práci boli materiály publikované na internete, najmä webové stránky jednotlivých implementácií JMS a nájdené existujúce porovnania. Cenným zdrojom boli tiež knihy zaoberajúce sa tematikou JMS, resp. komunikáciou aplikácií všeobecne.

Vytvorené riešenie (pozostávajúce z vybranej implementácie JMS a dvoch adaptérov) sa od septembra 2005 na Univerzite Komenského intenzívne používa na výmenu údajov o študentoch a ich preukazoch. Počas osemmesačnej prevádzky sme nezaznamenali väčšie problémy. Autor práce okrem vytvorenia samotného riešenia realizoval aj jeho inštaláciu na jednotlivých študijných oddeleniach, zaškolenie používateľov a vykonával priebežnú administráciu, monitorovanie systému a riešenie problémov. Testovanie poskytovateľov JMS, ako aj testovanie adaptérov pred ich uvedením do prevádzky bolo vykonané v Centre informačných technológií UK.

---

[1] Jedným z dôvodov zamerania sa na voľne dostupné produkty bolo, aby toto riešenie mohlo byť použité aj v organizáciách, ktoré nemajú prostriedky na obstaranie relatívne drahých komerčných riešení.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

CDO	centrálna databáza osôb
EAI	Enterprise Application Integration - proces integrácie aplikácií v rámci informačného systému organizácie
JDBC	Java Database Connectivity
JDK	Java Developer Kit
JMS	Java Message Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
MOM	Message Oriented Middleware - technológia umožňujúce asynchrónne posielanie správ medzi aplikáciami
PTP	doména point-to-point
PS	prístupový systém
pub/sub	doména publish/subscribe
UK	Univerzita Komenského v Bratislave
VIKUK	virtuálna knižnica Univerzity Komenského

0 Úvod

Na Univerzite Komenského (UK) sa vzhľadom na rýchly pokrok informačných technológií a vzrastajúce nároky na informačné zdroje v súčasnosti rieši úloha integrácie aplikácií v rámci jej informačného systému. Cieľom je dosiahnuť stav, že systémy pracujúce s informáciami budú mať tieto informácie k dispozícii a to bez nutnosti manuálneho prenášania údajov.

Vychádzajúc z [Chr00], informačným systémom organizácie nazývame množinu hardvérových a softvérových prostriedkov a pravidiel zabezpečujúcich zber, spracovanie a distribúciu informácií potrebných pre efektívny chod organizácie. Informačný systém organizácie sa skladá z jednotlivých subsystémov[2]. Subsystém alebo aplikácia pokrýva niektorú z oblastí činnosti organizácie, napr. študijná agenda, účtovníctvo, výučba a ďalšie. Integrovaný informačný systém je taký informačný systém, ktorého subsystémy sú navzájom prepojené. Integrácia podnikových aplikácií (Enterprise Application Integration) je proces integrácie aplikácií v rámci organizácie.

Jednou z úloh pri integrácii aplikácií je zaistenie komunikácie medzi týmito aplikáciami. Komunikácia môže byť riešená viacerými spôsobmi, ktoré sú podrobnejšie popísané v kapitole 2. Návrh riešenia. V práci sa bližšie venujeme jednému z nich, konkrétne riešeniu prostredníctvom Message Oriented Middleware (MOM). MOM označuje technológiu umožňujúcu asynchrónne posielanie správ medzi aplikáciami. MOM zbavuje programátora nutnosti riešiť problémy súvisiace s nedostupnosťou volanej aplikácie – zaslané správy čakajú v príslušnom rade realizovanom v MOM (v pamäti alebo na disku) dovtedy, kým nebude možné ich doručenie. Príslušné aplikácie sú takto zviazané voľnejšie ako v prípade synchrónnej komunikácie.

Existuje viacero produktov realizujúcich myšlienku MOM. Pre zaistenie portability aplikácií využívajúcich služby MOM bolo potrebné štandardizovať príslušné programátorské rozhranie (API). Java Message Service (JMS) predstavuje takýto štandard pre rozhranie, pre prístup k službe, pre vytvorenie, posielanie, prijímanie a čítanie správ. Je to všeobecne prijatý štandard, špecifikácia, ktorá definuje množinu rozhraní a im priradenú sémantiku. O implementáciu tohto štandardu sa pokúsilo viacero spoločností, ktoré vytvorili príslušné produkty. V ďalšom ich budeme označovať pojmami „implementácia JMS“, „poskytovateľ JMS“ (z angl. „JMS provider“), resp. jednoducho „produkt“.

[2] Pre časť informačného systému, ktorá je predmetom integrácie, budeme v práci používať pojmy „aplikácia“, „subsystém“, resp. tam, kde nehrozí nejednoznačnosť, aj pojem „systém“ (napríklad prístupový systém, systém ŠTUDENT, ...). Tieto tri pojmy budeme pre účely tejto práce považovať za synonymá.

Práca sa skladá z týchto častí:

- Prvá kapitola poskytuje informácie o informačnom systéme univerzity potrebné pre pochopenie problému, ktorým sa budeme zaoberať a stručný popis požiadaviek na komunikáciu medzi vybratými subsystémami.
- V druhej kapitole popíšeme existujúce prístupy ku komunikácii aplikácií a po ich porovnaní vyberieme najvhodnejší z nich pre riešenie komunikácie v rámci informačného systému UK.
- V tretej kapitole sú katalogizované požiadavky na produkt implementujúci JMS, na základe ktorých vyberieme najvhodnejší produkt.
- Štvrtá kapitola obsahuje informácie o existujúcich zverejnených porovnaniach produktov.
- V piatej kapitole sa snažíme vybrať relevantné nekomerčné implementácie JMS a porovnať ich vlastnosti na základe stanovených požiadaviek.
- Šiesta kapitola obsahuje test spoľahlivosti vybratých produktov, konkrétne ide o test korektnosti uloženia perzistentných správ vo výnimočných situáciách.
- V poslednej kapitole čitateľ nájde informácie o návrhu a implementácii adaptérov potrebných na zabezpečenie komunikácie použitím JMS medzi vybratými subsystémami informačného systému UK a o skúsenostiach počas prevádzky.
- Prílohy obsahujú popis rozhraní medzi centrálnou databázou osôb, adaptérom XCDO a systémom Študent a tiež používateľskú dokumentáciu k tomuto adaptéru.

1 Popis problému

Cieľom tejto práce je zaistenie výmeny údajov o študentoch univerzity medzi vybranými subsystémami informačného systému, ktorými sú študijný informačný systém (v ďalšom „systém Študent“, resp. len „Študent“), centrálna databáza osôb (CDO), virtuálna knižnica (VIKUK), prístupový systém (PS) a ďalšie.

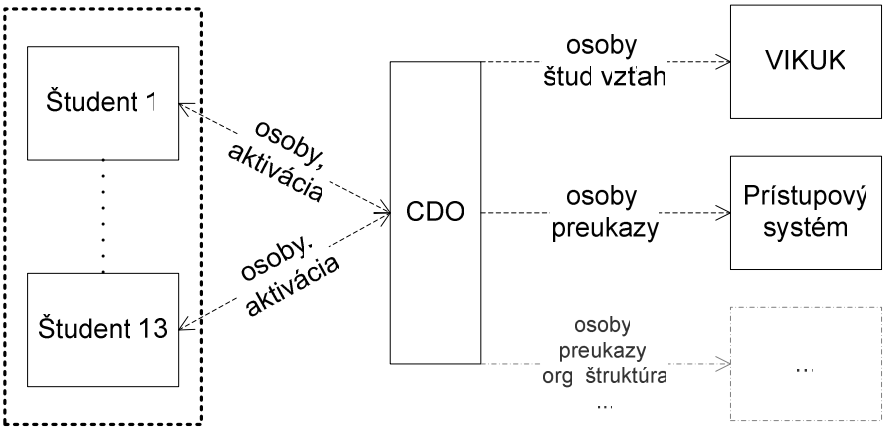
Systém Študent je prvotným zdrojom údajov o študentoch univerzity. Na každej z trinástich fakúlt je nasadená jedna jeho inštancia, oddelená od ostatných inštancií.

Údaje o študentoch je potrebné vhodným spôsobom centralizovať a poskytovať pre ďalšie využitie. Na uchovanie údajov získaných zo systému Študent slúži centrálna databáza osôb (CDO). CDO zabezpečuje sprostredkovanie údajov aplikáciám, ktoré ich potrebujú pre svoju činnosť. CDO zároveň preberá z databáz personálnych oddelení aj údaje o zamestnancoch, ale automatizácia tohto rozhrania nie je predmetom tejto diplomovej práce.

Virtuálna knižnica UK má dve hlavné obsahové oblasti. V prvej oblasti ide o automatizáciu klasických knižničných procesov ako sú napríklad: budovanie katalógu fakultných knižníc, výpožičky, spracovanie publikačnej činnosti a ďalšie. V druhej oblasti ide o zabezpečenie prístupu k externým informačným zdrojom (elektronické knihy, časopisy). Informácie o študentoch potrebuje na automatické vytváranie čitateľských kont.

Preukazy študentov a zamestnancov sa v celouniverzitnom rozsahu používajú na automatické povoľovanie vstupu oprávneným osobám použitím prístupového systému. Príkladom je umožnenie prístupu študentov do počítačových učební.

V práci hľadáme riešenie použiteľné pre komunikáciu všetkých (resp. aspoň väčšiny) subsystémov informačného systému UK, avšak pre jednoduchosť budeme konkrétne hovoriť o štyroch vyššie uvedených aplikáciách (resp. subsystémoch). Ich komunikácia je znázornená na nasledujúcom obrázku.



Obrázok 1-1: Komunikácia vybraných subsystémov na UK

Doteraz tieto systémy komunikovali tak, že potrebné činnosti (import a export údajov) zaisťovali manuálne ich správcovia. Táto komunikácia bola tým náročnejšia, čím viac systémov si potrebovalo medzi sebou vymieňať informácie. Preto bolo potrebné vytvoriť lepší spôsob a automatizovať čo najväčšiu časť spojení znázornených na obrázku. Pri každom zo spojení je znázornený obsah prenášaných údajov a smer komunikácie. Diagram zároveň šedou farbou naznačuje, že neboli spomenuté všetky systémy, len tie najdôležitejšie súvisiace s diplomovou prácou.

Cieľom je dosiahnuť stav, že systémy pracujúce s informáciami o študentoch budú mať tieto informácie k dispozícii, a to bez nutnosti ručného prenášania údajov a v priebehu niekoľkých hodín od pridania alebo modifikácie údajov v systéme Študent.

Podrobnejšie sa o jednotlivých systémoch a poskytovaných službách môžete dočítať v [KS06].

Každý systém je iný, napísaný v inom programovacom jazyku, bežiaci na inom operačnom systéme, používajúci inú architektúru, inú formu údajov, majúci iný spôsob prenosu a iné požiadavky na obsah údajov. Väčšina systémov je pripojená do pomerne rýchlej a spoľahlivej univerzitnej siete, niektoré sú chránené firewallom. Vo výnimočných situáciách je inštancia systému fyzicky odpojená od siete alebo pripojená pomalou linkou. Vek najstaršej aplikácie už prekročil pätnásť rokov a aj z tohto faktu vyplýva, že jej modifikácia pre naše účely je pomerne náročná. Pri riešení výmeny údajov je nutné zaistiť korektné spracovanie chybových stavov, spoľahlivosť, bezpečnosť, výkonnosť a dobrú prevádzkovateľnosť, pod ktorou máme na mysli jednoduchú administráciu, monitorovanie, hľadanie a odstraňovanie chýb, logovanie a upozorňovanie správcu na problémy. Na zaistenie spoľahlivosti je vhodné aspoň na strane CDO realizovať transakčné spracovanie príjmu, spracovanie a posielanie údajov. Pri úvahách o spoľahlivosti systému je potrebné počítať s výpadkami prúdu, zlyhaním operačného systému na pracovných staniciach, a zlyhaniami ľudského faktora.

Našou úlohou je vybrať vhodnú technológiu na zabezpečenie výmeny údajov medzi vyššie uvedenými systémami a následne implementovať a skúšobne nasadiť aplikácie potrebné na zabezpečovanie komunikácie medzi systémom Študent, CDO a VIKUK. Konkrétna realizácia doručovania údajov z CDO do PS nie je úlohou tejto diplomovej práce.

Pozrime sa podrobnejšie na subsystémy, ktorých komunikáciu máme zaistiť.

1.1 Centrálna databáza osôb (CDO)

CDO zabezpečuje:

- uchovávanie databázy všetkých študentov a zamestnancov univerzity,
- uchovávanie databázy všetkých vytlačených preukazov študenta a zamestnanca,
- aktivovanie preukazov študentov na základe informácií z fakúlt (pod aktiváciou rozumieme priradenie preukazu konkrétnemu študentovi, ktoré sa realizuje na študijnom oddelení),

- exportovanie potrebných údajov pre ďalšie systémy ako je virtuálna knižnica, prístupový systém a ďalšie.

Údaje o vytlačených preukazoch sa do CDO pridávajú vždy po tlači nových preukazov. CDO uchováva ich stav (aktívny, vrátený, stratený, ...). Je primárnym zdrojom týchto údajov.

Údaje o osobách sa do CDO nahrávajú pravidelne. Primárnym zdrojom údajov o študentoch je systém Študent. CDO uchováva len kópiu týchto údajov a pravidelne ich aktualizuje.

CDO je napísaná v programovacom jazyku Java. Vyvíjaná je v rámci univerzity.

1.2 Systém Študent

Systém Študent pomáha pri vykonávaní činností v oblasti študijnej agendy. S touto aplikáciou priamo pracujú zamestnanci študijných oddelení, členovia vedenia fakúlt a univerzity. Systém poskytuje funkcie týkajúce sa prijímacieho konania, zápisov, evidencie študentov, učiteľov, katedier a ostatných pracovísk fakúlt, študijných povinností a výsledkov, činnosti súvisiace s ukončením štúdia a ďalšie. Súčasťou systému je register študentov, ktorý uchováva a poskytuje údaje v zmysle zákona o vysokých školách.

Architektúra systému je klient/fileserver. Na každej fakulte je udržiavaná fakultná databáza. Centrálny register študentov je uložený na osobitnom súborovom serveri.

Aplikácia Študent je implementovaná v prostredí Clipper, pracuje v režime MS-DOS. Sú možné jej drobné úpravy, ako napríklad pridanie funkcie na uloženie potrebných údajov do súboru, ale nie je možné predpokladať, že by vedela priamo komunikovať s CDO. Táto funkčnosť môže byť obsiahnutá v špeciálnych aplikáciách, tzv. adaptéroch.

Fakultná inštancia systému Študent je nainštalovaná na pracovných staniciach s operačným systémom Windows 98, 2000, XP. Údaje sú uložené na fileserveri Windows NT 4.0, Windows 2000 alebo linux. Fileserver je pripojený do siete študijných oddelení, ktoré sú ďalej pripojené do univerzitnej siete, niekedy cez firewall. Na právnickej fakulte je sieť študijného oddelenia fyzicky oddelená od univerzitnej siete.

1.3 Virtuálna knižnica (VIKUK)

Ako bolo spomenuté vyššie, VIKUK slúži okrem iného na automatizáciu výpožičných procesov. Za účelom hromadného zakladania čitateľov – študentov je potrebné tejto aplikácii odovzdávať o nich niektoré informácie (minimálne meno, priezvisko, označenie fakulty a číslo preukazu). Číslo preukazu slúži ako identifikácia cez webovské rozhranie, pri vypožičiavaniach a ďalšie.

VIKUK využíva centralizovaný knižnično-informačný systém VIRTUA dodávaný spoločnosťou VTLS. Pracuje na serveroch IBM s operačným systémom AIX nad databázou Oracle.

1.4 Prístupový systém (PS)

Prístupový systém sa v súčasnosti používa na povoľovanie vstupu oprávneným osobám s použitím preukazov. Držiteľmi preukazov sú všetci študenti a zamestnanci univerzity. V prístupovom systéme jeho správcovia zaisťujú nastavovanie prístupových práv osôb a monitorovanie funkčnosti komponentov.

Prístupový systém okrem prevádzkovaných serverov pozostáva z dverových terminálov a koncentrátorov, ktoré zabezpečujú ovládanie týchto terminálov a ich komunikáciu s centrálnym databázovým serverom.

Prístupový systém pracuje na operačnom systéme MS Windows 2003 Server nad databázou MS SQL Server. Príslušná aplikácia je štandardným produktom, upraveným na zákazku pre UK.

Pozrime si, aké sú požiadavky na komunikáciu medzi vybranými subsystémami.

1.5 Komunikácia Študent-CDO

Údaje o študentoch sú do systému Študent vkladané pracovníkmi študijných oddelení. Odtiaľto je potrebné ich odovzdávať centrálnej databáze osôb. Z CDO prichádza systému Študent informácia o stave spracovania údajov, ktorú je potrebné zobraziť pracovníkom študijných oddelení. Výsledkom spracovania údajov z fakultného systému Študent môže byť poslanie správy o stave aj na iné fakulty (v prípade, že bola aktivovaná karta študenta študujúceho na viacerých fakultách).

Na strane CDO má byť príjem a spracovanie údajov automatické. Na strane systému Študent je potrebné zabezpečiť automatické posielanie údajov (v určených intervaloch), ale zároveň poskytnúť používateľom možnosť iniciovať komunikáciu ručne. V každom prípade musí mať používateľ k dispozícii informáciu o aktuálnom stave komunikácie.

Z topológie univerzitnej siete vyplýva, že treba vyriešiť aj prechod cez firewall, dokonca aj prenos údajov do inej siete, oddelenej od univerzitnej siete. V tomto prípade môžeme predpokladať, že v danej miestnosti je jedna pracovná stanica pripojená do univerzitnej siete a na druhej pracovnej stanici je nasadený systém Študent pripojený do vlastnej siete.

CDO funguje prakticky nepretržite, avšak nemôžeme predpokladať, že fakultné inštancie systému Študent budú vždy k dispozícii. Napriek tomu treba zaistiť, aby sa prenášané informácie za žiadnych okolností nestratili.

Ako už bolo spomenuté, treba sa zaoberať aj výnimočnými situáciami, ktoré sa môžu nastať počas prenosu údajov, bezpečnosťou prenosu, t.j. zachovaním autentickosti, integrity a dôvernosti prenášaných údajov.

1.6 Komunikácia CDO-VIKUK a CDO-PS

Úlohou je zaistiť pravidelný prenos údajov z CDO do VIKUK a PS (napríklad raz denne). Tieto údaje budú po doručení spracované cieľovým systémom. Opäť je potrebné zaistiť spoľahlivosť a bezpečnosť prenosu údajov. Predpokladá sa podobná komunikácia aj medzi inými subsystémami informačného systému univerzity.

Po objasnení situácie na UK, vysvetlení, čo chceme dosiahnuť a aké máme požiadavky na výmenu informácií ako celok aj na jeho vybraté komunikačné časti, pozrime si, aké základné techniky máme na riešenie daného problému k dispozícii.

2 Návrh riešenia

Pri integrácii aplikácií v rámci informačného systému organizácie (EAI) sa na zaistenie komunikácie najčastejšie používa tzv. middleware.

Middleware má viacero definícií. Konzorcium ObjectWeb [Obj05] používa nasledujúcu definíciu: V distribuovanom počítačovom prostredí je middleware softvérová vrstva, ktorá leží medzi operačným systémom a každou z aplikácií. V tomto texte budeme pod middleware rozumieť softvér uľahčujúci komunikáciu medzi programami v distribuovanom prostredí.

Middleware pozostáva spravidla z:

- komunikačného mechanizmu (volanie procedúr, operácií, zasielanie správ),
- množiny podporných služieb (bezpečnosť, adresárové služby, podpora transakcií, zdieľanie záťaže, ...).

Middleware by mal poskytovať nasledovné vlastnosti:

- ľahkosť použitia (v porovnaní s vytvorením vlastného riešenia pre komunikáciu založeného na prostriedkoch nižšej úrovne ako napr. BSD Sockets),
- transparentné umiestnenie (aby aplikácie nepotrebovali vedieť o sieti a o adrese druhej strany a správali sa, akoby boli na jednom počítači),
- zaistenie, že údaje nemôžu byť stratené alebo viacnásobne doručené,
- zaistenie, že údaje sa nesmú byť poškodené,
- transparentnosť programovacieho jazyka (aby si aj aplikácie napísané v rôznych programovacích jazykoch vedeli medzi sebou vymieňať informácie).

Integrita posielania údajov (t.j. údaje nesmú byť stratené, viacnásobne doručené alebo poškodené; bezpečnostná integrita nie, tá patrí medzi vlastnosti vyššej úrovne) je obvykle riešená sieťovým softvérom ako je TCP/IP. Väčšina technológií middleware poskytuje transparentnosť umiestnenia.

Čitateľ môže nájsť ďalšie informácie o middleware napr. v [Chr00], odkiaľ boli čerpané predošlé poznámky.

Na to, aby sme vedeli navrhnúť riešenie problému komunikácie aplikácií, potrebujeme poznať základné techniky na riešenie takýchto situácií. Na tieto účely slúžia takzvané integračné štýly, ktoré opíšeme v nasledujúcej podkapitole.

2.1 Integračné štýly

Nasledujúce informácie o integračných štýloch boli čerpané z [Gre04], pričom sme pri ich spracovaní zohľadnili situáciu na Univerzite Komenského.

Keby boli požiadavky na integráciu vždy rovnaké, bolo by možné vo všetkých prípadoch použiť rovnaký spôsob riešenia. Žiaľ, situácia nie je taká jednoduchá. Pri rozhodovaní sa môžeme stretnúť s nasledujúcimi kritériami – pri každom integračnom projekte môže byť dôležitosť jednotlivých kritérií odlišná:

- Zviazanosť aplikácií – minimalizáciou zviazanosti môže každá aplikácia poskytovať svoje služby bez toho, aby pri jej zlyhaní prestali pracovať aj ostatné s ňou prepojené aplikácie (napr. pri zlyhaní CDO nechceme, aby prestali fungovať systémy Študent a VIKUK).
- Nutnosť zmien (intrusiveness) – pri integrácii existujúcich aplikácií sa vo všeobecnosti snažíme minimalizovať zmeny kódu týchto aplikácií. Úpravy sú problematické najmä pri starších aplikáciách, ako je napr. systém Študent a takisto pri aplikáciách dodaných externým dodávateľom, ako je napr. VIKUK.
- Včasnosť dát (data timeliness) – snažíme sa minimalizovať čas potrebný na prenos údajov medzi aplikáciami.
- Údaje alebo funkcie – niektoré integračné riešenia podporujú nielen zdieľanie údajov, ale aj prístup aplikácie k funkciám iných aplikácií (napríklad volanie vzdialených procedúr).
- Spoľahlivosť – údaje sa za žiadnych okolností nesmú stratiť.

Vidíme, že máme veľa rôznych kritérií, ktoré by mali byť stanovené a bolo by vhodné povedať, ktoré kritériá sú zaujímavé pre nás pri výbere integračných prístupov. Kritériá pre potreby UK z týchto sú: minimalizácia zviazanosti aplikácií a nutnosti zmien, prenos údajov bez nutnosti volania funkcií a spoľahlivosti. Včasnosť dát v súčasnosti nie je kritická (t.j. pár minút nám nerobí problémy).

Naďalej zostáva otázne, ktoré integračné štýly najlepšie spĺňajú ktoré kritériá. Budeme sa zaoberať nasledujúcimi integračnými štýlmi:

- Prenos údajov cez zdieľané súbory (File Transfer) – aplikácia produkuje súbory s údajmi, ktoré ostatné aplikácie čítajú.
- Zdieľaná databáza (Shared Database) – aplikácie ukladajú údaje na zdieľanie do spoločnej databázy.
- Volanie vzdialených procedúr (Remote Procedure Invocation) – každá aplikácia ponúka svoje procedúry, ktoré môžu ostatné aplikácie (na diaľku) volať. Takto je možné realizovať aj výmenu údajov.
- Posielanie správ (Messaging) – komunikujúce aplikácie si vymieňajú údaje (a prípadne volajú svoje funkcie) posielaním správ realizovaným prostredníctvom spoločného systému na doručovanie správ.

Pozrime sa podrobnejšie na jednotlivé integračné štýly.

2.2 Prenos údajov cez zdieľané súbory

Prenos údajov cez zdieľané súbory funguje tak, že každá aplikácia produkuje súbor s údajmi, ktoré chce poskytovať ostatným aplikáciám (tento proces nazývame aj

export údajov). Tento súbor využije následne druhá aplikácia, ktorá tieto údaje potrebuje (načítanie a spracovanie údajov nazývame aj importom). Prenos súborov môžeme považovať za najjednoduchšie riešenie pre middleware[3].

Súbory sú univerzálne pamäťové mechanizmy, prítomné prakticky v každom operačnom systéme a dostupné pre ľubovoľný programovací jazyk.

Otázkou je, kedy súbory produkovať a konzumovať. Vzhľadom na určité množstvo úsilia potrebné na produkovanie a spracovanie súborov obvykle nechceme túto operáciu vykonávať príliš často. Následné problémy synchronizácie nemusia byť vážne, avšak v niektorých prípadoch je to kritický bod. Predstavme si napríklad, že v stredu v knižnici prídu na to, že študentka zabudla vrátiť jednu knihu. Predtým sa študentka odsťahovala a v pondelok zmenila adresu na študijnom oddelení. V prípade, že posielanie údajov medzi systémami prebieha raz týždenne, študentka by sa o upomienke nedozvedela[4]. Ak by sme synchronizovali údaje aspoň denne, vyhli by sme sa viacerým takýmto situáciám spôsobeným nekonzistenciou údajov.

Samozrejme nie je dôvod, prečo by sme nemohli súbory produkovať častejšie. V skutočnosti môžeme považovať prenos údajov cez zdieľané súbory za formu komunikácie posielaním správ a prenášať tieto súbory aj veľmi často – vždy, keď nastane zmena v aplikácii. Problémom je potom manažovanie všetkých súborov, ktoré boli produkované tak, aby bolo zabezpečené, aby každý bol prečítaný, a aby sa ani jeden nestratil.

Veľká výhoda súborov je, že integrátori (ľudia, ktorí sa snažia zabezpečiť, aby aplikácie spolupracovali) nepotrebujú žiadne vedomosti z vnútra aplikácie. Aplikačné tímy obvykle samy poskytujú možnosť exportu súboru. Súbory sa takto stávajú verejnými rozhraniami medzi aplikáciami. Túto vlastnosť používame aj pri systéme Študent.

V prípade zdieľanej databázy a volania vzdialených procedúr naznačíme len výhody a nevýhody voči predošlým riešeniam, lebo sa nimi nebudeme ďalej zaoberať. Ďalšie podrobnosti čitateľ nájde v [Chr00] a v [Gre04].

2.3 Zdieľaná databáza

V tomto prípade používajú jednotlivé aplikácie na ukladanie svojich údajov jednu, zdieľanú databázu, ku ktorej má každá z nich prístup. Takto je zabezpečené, že v prípade zmeny sú najaktuálnejšie údaje okamžite viditeľné aj v ostatných aplikáciách.

Výhody zdieľanej databázy oproti výmene údajov prenosom súborov:

---

[3] Napríklad s použitím fileservera a mapovaním toho istého sieťového disku na dva počítače umiestnené na dvoch miestach siete vieme zabezpečiť asi najjednoduchšie riešenie integrácie aplikácií. Dve aplikácie budú komunikovať tak, že jedna zapíše potrebné údaje do súboru do namapovaného adresára a keďže ten istý adresár je vidno aj na druhom počítači, druhá aplikácia môže údaje priamo prečítať a spracovať.

[4] Samozrejme, tento príklad je len ilustračný (pravdepodobne jej noví majitelia/nájomníci upomienku doručia).

- okamžitá aktualizácia, z čoho vyplýva zmenšená nekonzistencia údajov,
- systém riadenia transakcií (pomôže správne organizovať aktualizáciu),
- jednoduchý prístup k databáze cez SQL.

Nevýhody:

- v prípade integrácie existujúcich aplikácií sa vyžaduje ich zmena (prispôsobenie sa tomu, že spoločné údaje budú v centrálnej databáze), príkladom je systém Študent,
- výrobcovia externých aplikácií si spravidla vyhradzujú právo na zmenu schémy databázy v novších verziách,
- ťažké je navrhovanie zdieľanej databázy, konkrétne najmä dosiahnutie jednotnej schémy,
- je možnosť uviaznutia (jedna transakcia čaká na údaje, ktoré druhá zamkla a tá druhá čaká na údaje, ktoré prvá zamkla ešte pred ňou), navyše zamykanie údajov môže značne spomaliť prístup k databáze.

2.4 Volanie vzdialených procedúr

Volanie vzdialených procedúr umožní počítačovým programom bežiacim na jednom počítači zavolať podprogram na druhom počítači bez toho, aby sa programátor musel zaoberať detailmi kvôli tejto interakcii.

Výhody volania procedúr voči zdieľanej databáze a prenosu údajov cez zdieľané súbory:

- okrem prenosu údajov môžeme priamo vykonať aj akcie,
- ľahšia reakcia na zmeny aplikácií ako pri zdieľanej databáze, kde zmeniť schému je náročné a jej zmena môže spôsobiť zmeny vo viacerých aplikáciách (napr. v našom prípade by v prípade zmeny databáz v CDO bolo treba zmeniť aj systém Študent),
- existuje veľa podporujúcich technológií (CORBA, DCOM, .NET Remoting, Java RMI); v čase písania diplomovej práce boli obľúbené webové služby, ktoré pracujú cez HTTP, a tak nemajú problémy s prechodom cez väčšinu firewallov. Ďalšie podrobnosti o webových službách sa môžete dočítať v diplomovej práci [Bis06].

Nevýhody:

- vzdialené volanie vedie k vyššej zviazanosti komunikujúcich aplikácií,
- v prípade systému Študent nie je zabezpečená táto technológia a bolo by ju treba doprogramovať,
- počas volania vzdialených procedúr je klientska aplikácia blokovaná, preto je vhodné použiť dve vlákna, pričom jedno sleduje vstupy (klávesnica a myš) a druhé vykonáva volanie vzdialených procedúr. Písanie viacvláknových aplikácií je náročnejšie a nie je možné vo všetkých prostrediach (napr. ani v jazyku Clipper, v ktorom je vytvorený systém Študent).

2.5 Posielanie správ

Prenos údajov cez zdieľané súbory a vzdialenú databázu umožňuje zdieľať údaje, ale nie funkčnosť. Volanie vzdialených procedúr umožňuje zdieľať funkčnosť za cenu vytvorenia silnej väzby medzi aplikáciami.

Pre integráciu aplikácií sa výhodný zdá byť prenos údajov cez zdieľané súbory, kde môžeme vyrobiť veľmi rýchlo mnoho malých údajových balíkov, ľahko ich preniesť a zaistiť, aby bola prijímajúca aplikácia automaticky informovaná o tom, že je k dispozícii nový balík údajov. Prenos potrebuje mechanizmus zaisťujúci úspešnosť aj pri chybách v komunikácii. Detaily diskovej štruktúry alebo databázy pre ukladanie údajov by mali byť pred aplikáciami skryté, aby bolo možné jednotlivé detaily v prípade potreby ľahko zmeniť. Prenos údajov by mal byť asynchrónny, aby vysielajúca strana nemusela čakať na konzumenta (s výnimkou prípadu, keď potrebuje čakať na odpoveď). Takto je možné vyhnúť sa šíreniu chýb v systéme.

Analógiou RPC je telefónny rozhovor, analógiou pri posielaní správ je veľmi rýchla poštová služba, ktorou môžete posielať správy aj vtedy, keď konzument alebo sieťové spojenie nie je k dispozícii. Middleware zabezpečuje prenos cez rady (queues), zaisťuje, že v prípade zlyhania siete sa správy nestratia, a navyše zabezpečuje doručovanie údajov práve raz. Podobá sa na zápis a čítanie z TCP/IP socketu, ale sú tu nasledujúce kľúčové rozdiely:

- rady majú mená,
- rady sú nezávislé na komunikujúcich aplikáciách (ktoré by využili socket), preto viacero aplikácií môže naraz posielať do toho istého radu,
- ak sieť zlyhá, správy môžu čakať v rade, v prípade potreby je možné ich perzistentne ukladať napríklad na pevný disk, aby sme sa vyhli strate údajov pri zlyhaní programu. Tieto požiadavky už socket nedokáže priamo zabezpečiť,
- na radoch je možné vykonať transakciu obsahujúcu vybratie niekoľkých správ a vykonanie databázových úprav. V prípade nutnosti je automaticky zabezpečené vykonanie rollback, aby správy zostali v rade a súčasne sa vykonal rollback nad databázou.

Posielanie správ znižuje zviazanosť aplikácií, podobne, akoby sme používali prenos údajov cez zdieľané súbory. Integrátori môžu posielať správy k viacerým konzumentom naraz, smerovať správy jednému z množiny konzumentov, alebo použiť ďalšie topológie. Toto oddeľuje integračné rozhodnutia od vývoja aplikácií.

S častým posielaním malých správ môžeme dovoliť aplikáciám väčšiu spoluprácu. Informácie môžeme požiadať a odpoveď môže byť rýchla. Síce takáto spolupráca aplikácií nie je taká rýchla ako je volanie vzdialených procedúr, vyvolajúci sa nemusí zastaviť a čakať, kým správa je spracovaná a odpoveď je prijatá (t.j. asynchrónne spracovanie). Posielanie správ má veľa riešení, z ktorých dokážu vybaviť tisícky správ za sekundu.

Keďže sa budeme zaoberať technikou posielania správ, mohli by sme predpokladať, že ide o najlepšie riešenie na integráciu podnikových aplikácií.

Vo všeobecnosti to nemusí byť pravda. Zvýšenie frekvencie pri posielaní správ redukuje problémy s nekonzistentciou, ktoré sa vyskytovali pri prenose údajov cez zdieľané súbory, ale neodstraňuje ich celkom. Stále sa môžu vyskytnúť chyby spôsobené nie úplne plynulou aktualizáciou. Zároveň testovanie a ladenie aplikácií v asynchrónnom prostredí je ťažšie, než v synchrónnom.

Nevýhody posielania správ sú, že správy sú postupnosti za sebou idúcich bitov, a je už len na integrátoroch, aby zaistili, že producent a konzument správ sa zhodli na ich štruktúre. Ďalšou nevýhodou je, že rady sa nachádzajú na strane servera a keď sieť zlyhá medzi vysielajúcou stranou a serverom, správy nie je možné doručiť. Niektoré produkty však riešia aj túto situáciu ukladaním správ do dočasného radu na strane klienta, alebo zabezpečením zotavenia z poruchy bez toho, aby musel byť klient o tom informovaný.

S posielaním správ sa môžeme stretnúť ako súčasťou technológií, ako je J2EE. V čase písania diplomovej práce bola za najperspektívnejšiu technológiu v rámci riešenia integrácie aplikácií považovaná podniková zbernica služieb (Enterprise Service Bus, ESB), ktorej základom je práve posielanie správ (messaging). Aj z týchto faktov môžeme usudzovať, že myšlienka komunikácie posielaním správ je perspektívnou a má zmysel sa ňou zaoberať.

Podrobnejšie o integračných štýloch a vzoroch pre EAI nájdete v [Gre04], o ESB môžete sa dočítať v [Ter06].

Pri vyberaní vhodného integračného štýlu je odporúčané nepoužívať všade jeden štýl, ale vybrať si štýl najvhodnejší pre dané potreby. Každý štýl má svoje výhody a nevýhody. Aplikácie môžeme integrovať použitím viacerých štýlov tak, aby každý bod integrovania využil to najlepšie riešenie. Následkom tohto je, že veľa prístupov k integrácii môžeme vidieť ako spojenie viacerých integračných štýlov.

Sústredíme sa teda na posielanie správ, a to preto, lebo tento princíp považujeme za vhodný vzhľadom na naše integračné požiadavky. V rámci komunikácie systému Študent s CDO budeme používať kombináciu dvoch integračných štýlov: prenos údajov cez zdieľané súbory a posielanie správ.

2.6 Java Message Service

Informácie k tejto boli čerpané hlavne z [Sun02] a z [Sha02]. Budeme sa zaoberať špecifikáciou Java Message Service (ďalej len JMS) verzie 1.1 zo dňa 12. apríla 2002.

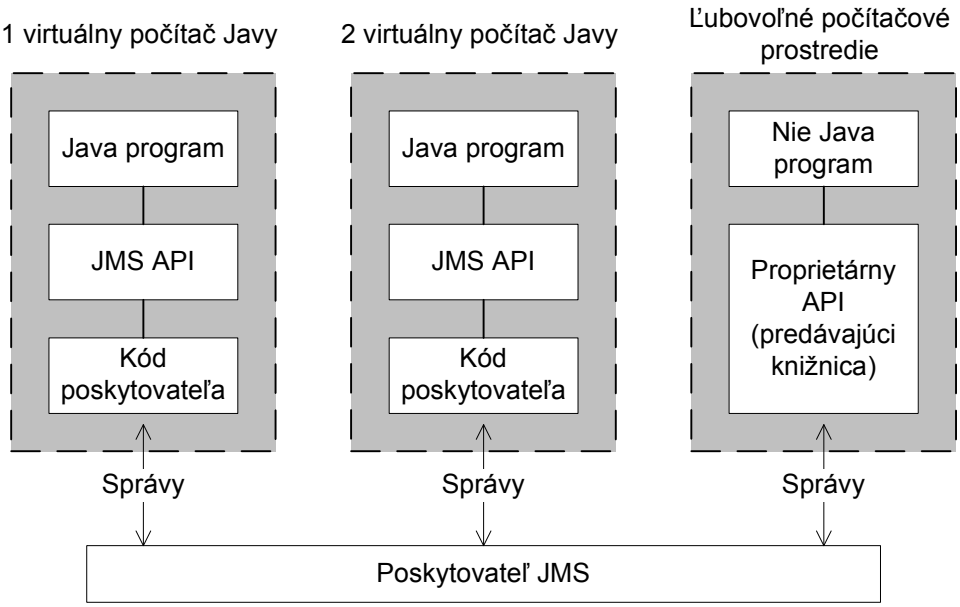
Java Message Service poskytuje nástroj, ktorým môžu programy v jazyku Java vytvárať, posielať, prijímať a čítať správy. JMS predstavuje množinu rozhraní a k nim priradenej sémantiky. Správy sú údaje, požiadavky, hlásenia alebo simulované udalosti, ktoré sú vytvorené a následne konzumované podnikovými aplikáciami.

JMS je špecifikácia a nie produkt. Je výsledkom snahy spoločnosti Sun Microsystems a jej partnerov na vytvorenie priemyselného štandardu rozhrania aplikačných programov (API), ktorý pokrýva niekoľko základných typov posielania správ medzi aplikáciami. Špecifikácia JMS je voľne prístupná prostredníctvom internetu, viď. [Sun02].

Implementáciu JMS rozhrania budeme nazývať aj poskytovateľ JMS (z angl. JMS provider). JMS bol navrhnutý tak, aby bolo možné bez zmeny a znovukompilovania vymeniť jednu knižnicu za druhú, t.j. aplikáciu, ktorú používa len JMS API je možné použiť pri všetkých poskytovateľoch spĺňajúcich špecifikáciu JMS.

Hlavné obmedzenie je, že k JMS API je možné pristupovať len z programu napísaného v Jave. Niektorí poskytovatelia JMS však obsahujú knižnice aj na prístup z iných programovacích jazykov (napr. C++, C#, ...).

JMS nič nehovorí o tom, akým spôsobom majú byť údaje prenášané. To je už len na poskytovateľoch JMS, aký protokol na komunikáciu používajú (napr. TCP, SOAP, HTTP, ...). Komunikáciu medzi programami napísanými v jazyku Java, resp. v iných programovacích jazykoch môžeme znázorniť nasledovne:



Obrázok 2-1: Spolupráca medzi poskytovateľom JMS, Java programami a nie Java programom

Pri komunikácii prostredníctvom JMS nie je potrebné, aby aplikácie boli spustené na jednom počítači, ani realizované na jednej architektúre. JMS bol navrhnutý na doručenie správ medzi aplikáciami, ktoré o sebe navzájom vedia málo, prípadne vôbec nič. T.j., aplikácie (klienti JMS) nikdy nekomunikujú medzi sebou priamo cez sieť, ale cez ďalší samostatne bežiaci program (poskytovateľ JMS, server), ktorý má za úlohu zabezpečiť prenos správ.

2.6.1 Správa JMS

Správa JMS obsahuje nasledovné časti:

- záhlavie (Header) – pozostáva z nutných hodnôt (polí), ktoré používajú klienti aj poskytovateľ JMS na identifikovanie a smerovanie správ,
- vlastnosti (Properties) – doplňuje záhlavie, poskytuje zabudované prostredie na voliteľné polia. Obsahuje v sebe:
  - vlastnosti špecifické pre danú aplikáciu (klienta)
  - štandardné vlastnosti (napríklad: JMSRedelivered, JMSDeliveryMode)

- a špecifické vlastnosti poskytovateľa JMS,

- telo (Body) – slúži na ukladanie údajov.

Pri doručení správy poskytovateľ JMS má za úlohu nastaviť vlastnosť JMSRedelivered na pravdivú (true), ak sa pokúša správu doručiť klientovi opakovane. Správa obsahuje aj vlastnosť na určenie typu doručenia (JMSDeliveryMode), ktoré môže byť:

- perzistentné (PERSISTENT) alebo
- neperzistentné (NON_PERSISTENT).

Správy JMS označené ako perzistentné sú tie, ktoré sa za žiadnych okolností nesmú stratiť, t.j. po doručení k poskytovateľovi JMS musia byť na strane servera trvalo uložené (napríklad na disk alebo do externej databázy).

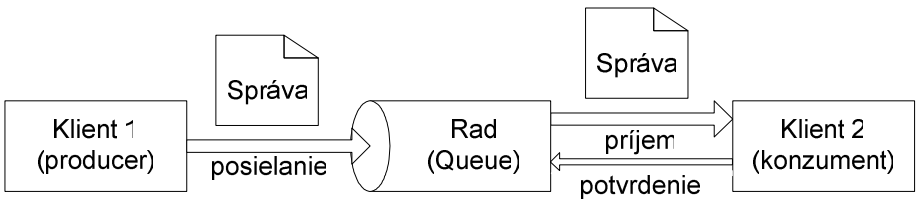
Telo správy JMS môže obsahovať jeden z nasledovných:

- reťazec – java.lang.String (TextMessage),
- prúd neinterpretovaných bajtov (BytesMessage),
- serializovateľný Java objekt (ObjectMessage),
- množinu párov: meno–hodnota, kde meno je reťazec a hodnota je hodnotou zo základných dátových typov Java (MapMessage),
- prúd Java primitív, písaný a čítaný sekvenčne (StreamMessage),
- alebo nič, t.j. prázdna správa (empty message).

Ďalšou dôležitou vlastnosťou je určenie cieľa (Destination), kde bude správa doručená, ktorá úzko súvisí so zvolenou doménou. Podporované domény sú point-to-point a publish/subscribe.

2.6.2 Doména „point-to-point“

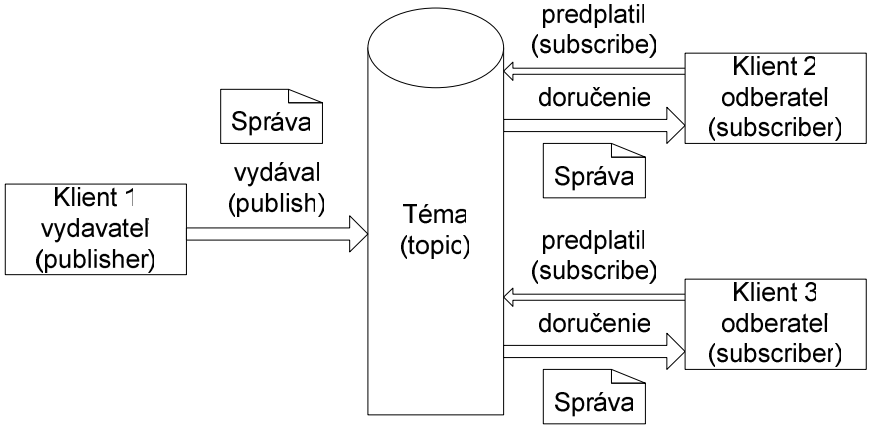
Doména „point-to-point“ (PTP) je postavená na myšlienke radu. Producent (odosielateľ) pošle príslušnú správu do vybratého radu (cieľa/Destination) a poskytovateľ JMS ju uchová, kým ju nie je možné doručiť. Keď je konzument dostupný, vyberá správy z prislúchajúceho radu a po úspešnom doručení a spracovaní správu potvrdzuje. Každá správa má len jedného konzumenta, ale do jedného radu môžu posielať správy aj viacerí producenti. Používanie domény PTP je vhodné, keď každá správa musí byť spracovaná práve jedným konzumentom. Situácia je znázornená na nasledujúcom obrázku:



Obrázok 2-2: Znázornenie komunikácie v doméne point-to-point

2.6.3 Doména „publish/subscribe“

V doméne „publish/subscribe“ (pub/sub) vydavateľ (publisher) pošle správu do témy (topic, analógiou je rad v PTP). Správu dostanú všetci aktívni odberatelia (subscriber) prihlásení na odber danej témy. Neprihlásení klienti a pasívni odberatelia správu nedostanú. Výnimku tvoria len trvalí (durable) predplatitelia, kde sa poskytovateľ JMS postará o ukladanie správ, kým sa nestanú aktívnymi. V prípade domény pub/sub je zaručené, že nielen viacero klientov môžu poslať správu do témy, ale správa môže byť doručená aj viacerým klientom naraz. Používanie domény pub/sub je vhodné, keď každá správa môže byť spracovaná 0, 1 alebo viacerými klientmi. Komunikácia je znázornená na nasledujúcom obrázku:



Obrázok 2-3: Znázornenie komunikácie v doméne publish/subscribe

Keď chceme zabezpečiť, aby správa bola doručená práve raz, treba použiť posielanie perzistentných správ a trvalé pripojenie súčasne. Táto metóda je najpomalšia, ale aj najspoľahlivejšia. Ďalšie spôsoby doručenia znázorňuje nasledujúca tabuľka (v prvom stĺpci je spôsob doručenia správ k poskytovateľovi JMS a v prvom riadku je spôsob odberania):

	Netrvalý odberateľ (non-durable subscriber)	Trvalý predplatiteľ
NON_PERSISTENT	nanajvýš raz (chyba ak je neaktívny)	nanajvýš raz
PERSISTENT	práve raz (chyba ak je neaktívny)	**práve raz**

Tabuľka 2–1: Spôsoby doručenia v doméne pub/sub

2.6.4 Potvrdzovanie správ

Na zvýšenie spoľahlivosti doručenia správ slúži mechanizmus potvrdzovania prijatia správ, ktorým prijímateľ signalizuje poskytovateľovi JMS, že správu úspešne dostal, resp. spracoval. Poznáme tri typy potvrdzovania:

- DUPS_OK_ACKNOWLEDGE – potvrdzovanie je zabezpečené automaticky dávkovo (napríklad po doručení desiatich správ je potvrdenie poslané pre všetky správy naraz), preto sa môže stať, že správy budú doručené aj druhýkrát po zlyhaní . Je to najrýchlejšie riešenie (zmenšený overhead).
- AUTO_ACKNOWLEDGE – automaticky potvrdzuje každú správu po jej úspešnom doručení klientovi.

- CLIENT_ACKNOWLEDGE – najspoľahlivejšia potvrdzovacia metóda dovoľuje potvrdenie vyvolať klientskym programom. Takto je možné zabezpečiť, že je vykonané nielen doručenie správy, ale aj jej spracovanie. Nevýhodou je, že v tomto prípade klient musí zavolať sám príslušnú metódu a v ostatných prípadoch to bolo realizované bez vedomia programátora.

2.6.5 Lokálne transakcie

Ďalšiu úroveň spoľahlivosti môžeme zabezpečiť, keď použijeme lokálne transakcie. V prípade asynchrónneho spracovania nemôžeme použiť v jednej transakcii posielanie správ k poskytovateľovi JMS a príjem správ druhým klientom, lebo tak by sme stratili asynchrónnosť (producent by čakal v spracovaní transakcie pokiaľ nedokončí konzument svoju prácu, t.j. prenos by sa stal synchrónnym).

Transakcie je vhodné použiť napríklad keď potrebujeme prijať niekoľko správ, spracovať ich a následne poslať niekoľko odpovedí (v rámci transakcie môžeme poslať správu aj do iného radu, resp. témy).

2.6.6 Distribuované transakcie (XA support)

V niektorých prípadoch nám však nestačia ani lokálne transakcie. Keď potrebujeme spojiť okrem lokálnych transakcií v rámci JMS aj iné transakcie (napríklad transakcie obsahujúce aj databázové úpravy) do jedného celku, potrebujeme XA (javax.transaction.xa.XAResource), aby sme vedeli využiť služby externého správcu transakcií (Transaction Manager). XAResource rozhranie je založené na štandarde od Open Group (Distributed Transaction Processing: The XA Specification). Toto používa aj špecifikácia Java Transaction API (JTA). Podrobnosti o špecifikácii XA sa môžete dočítať v [XOp91], o Open Group [Ope06], o JTA [JTA02].

Jednoduchším riešením je použiť message-driven bean, kde sa programátor nemusí zaoberať špeciálnym kódom týkajúcim sa spojenia spracovania správ a databázových úprav. Podrobnejšie o message-driven bean sa môžete dočítať napríklad v [Ric01]. Na podrobnejšie štúdium o použití distribuovaných transakcií v rámci JMS použitím databázových úprav odporúčame [Sha02]. Touto témou sa podrobnejšie nebudeme zaoberať, lebo používanie message-drive bean je využiteľné na strane CDO, ktorou sa zaoberá [Ter06].

2.6.7 Kedy nepoužívať JMS

JMS môžeme používať v doméne pub/sub alebo v point-to-point, existujú však situácie, v ktorých toto použitie nie je vhodné. Napríklad na posielanie e-mailov je už dobre podporovaný priemyselný štandard SMTP, IMAP a ďalšie. Streaming audio/video potrebuje spojový prenos (connection-oriented), preto ani v tomto prípade nie je JMS vhodné riešenie.

Taktiež JMS nie je vhodné na hromadnú komunikáciu prostredníctvom siete internet. Nie je to chyba JMS API, ide skôr o obmedzenia internetu brániace použiť túto možnosť. Podrobnosti možno nájsť v [Sha02].

3 Požiadavky na poskytovateľa JMS

IEEE-Std 830 – 1993 [IEE93] popisuje zoznam nefunkčných požiadaviek, ktoré by bolo dobré zahrnúť do dokumentu o softvérových požiadavkách. Podľa tohto zoznamu boli špecifikované aj naše požiadavky na poskytovateľa JMS. (V našom prípade síce nejde o špecifikáciu požiadaviek na novo vytváraný systém, napriek tomu je IEEE-Std 830 – 1993 vhodnou inšpiráciou pre štruktúru tohto dokumentu.)

Požiadavky na funkcie:

1. Poskytovateľ JMS musí podporovať synchrónny aj asynchrónny príjem správ (táto požiadavka je súčasťou špecifikácie JMS). 5
2. Poskytovateľ JMS musí vedieť pracovať minimálne s údajmi typu XML a binárnym súborom (súčasť špecifikácie JMS, typy správ: TextMessage, BytesMessage).
3. Podporované musia byť obe domény (v špecifikácii JMS je nutná podpora aspoň jednej z nich):
   a. point-to-point,
   b. publish/subscribe.
4. Vítame[6]: poskytovateľ JMS by mal podporovať distribuované transakcie (odporúčané v špecifikácii JMS ako „XA support“), plánuje sa využitie tejto črty na strane CDO.

Požiadavky produktu týkajúce sa obmedzenia prostredia

5. Komunikácia musí byť umožnená prostredníctvom štandardnej TCP/IP siete
   a. aj pri rýchlosti 128kbps.
6. Musí byť umožnený prechod cez firewall (napr. podpora protokolov http, https), alebo možnosť jednoduchej konfigurácie v bežných firewalloch (napr. použitie TCP/UDP portu s pevným, a nie variabilným číslom).
7. Poskytovateľ JMS (server) musí pracovať aj na bežnom počítači dostupnom počas písania diplomovej práce[7].

[5] Synchrónnym príjmom správ sa rozumie synchrónna komunikácia medzi poskytovateľom JMS a klientom, t.j. preberanie správ funkciou consumer.receive(...), asynchrónnym príjmom volanie klientom pripravenej funkcie onMessage(...). Samozrejme, komunikácia medzi producentom a konzumentom správ je vždy asynchrónna.
[6] vítaná, ale nie je to nutnosťou

8. Poskytovateľ JMS musí byť schopný pracovať na operačnom systéme Windows Server 2003 a Linux.

Požiadavky použiteľnosti (usability)

9. Poskytovateľ JMS musí byť schopný poskytovať vhodné prostriedky na manažment, monitorovanie, prevádzkovanie.
10. Poskytovateľ JMS musí obsahovať kvalitnú dokumentáciu, a musí existovať možnosť podpory (napr. fórum, hotline, ...).

Požiadavky spoľahlivosti (reliability)

11. Správy sa za žiadnych okolností nesmú stratiť (napr. ani pri výpadku prúdu).
12. Vítame: poskytovateľ JMS by mal obsahovať prostriedky pre zaistenie vysokej dostupnosti (High Avaliability).

Požiadavky na bezpečnosť (security)

13. Musí byť zaistené, že správu do príslušného radu vložil naozaj adaptér prislúchajúci danému systému (autentickosť).
14. Musí byť zaistené, že správu si môže prevziať len oprávnený konzument správ (autorizácia).
15. Komunikácia pri prenose bude chránená proti odpočúvaniu a neoprávnenej modifikácii (integrita, napr. prostredníctvom SSL).

Požiadavky na využitú kapacitu (capacity)

16. Poskytovateľ JMS musí byť schopný bez problémov:
    a. prijať správy veľkosti až 10MB (VILI),
    b. ukladať správy doručené do jedného týždňa, t.j. cca 2*5*13*1MB = 130 MB,
    c. uchovať správy pre VILI jeden mesiac, t.j.: 20*10MB = 200 MB.
17. Vítame: funkcie by mali byť prístupné aj v iných programovacích jazykoch.

Požiadavku na štandardizáciu (standard)

18. Poskytovateľ JMS musí spĺňať špecifikáciu JMS ver 1.1.

Dodávacie požiadavky (delivery)

19. Poskytovateľ JMS musí byť voľne sťahovateľný z internetu, kde je vždy najnovšia dostupná stabilná verzia spolu s aktuálnou a podrobnou administrátorskou a programátorskou dokumentáciou.

---

[7] t.j. napr. na Intel Celeron CPU 2 GHz, 512 MB RAM, 80 GB HDD – aby bolo umožnené testovanie produktu

Požiadavka na spoluprácu t.j. interoperabilita (interoperability)

20. Vítame: pri ukladaní prezistentných správ do databázy JDBC.

Výkonnostné požiadavky (performance)

21. Nepredpokladajú sa veľké objemy spracúvaných správ:
    a. správy obsahujúce celú databázu: denne do cca 150 správ s maximálnou veľkosťou 1MB,
    b. správy obsahujúce len zmeny: denne do cca 100 000 správ s maximálnou veľkosťou 1kB.

Po špecifikovaní požiadaviek sa pozrime na existujúce porovnania s cieľom získať informácie relevantné pre výber implementácie JMS vhodnej pre potreby informačného systému UK.

4 Existujúce porovnania

Pri hľadaní porovnaní existujúcich implementácií JMS špecifikácie sme našli dokument [Nat01] zverejnený v roku 2001 venovaný architektonickej analýze middleware zameraného na posielanie správ (MOM). V tomto dokumente sa nachádza porovnanie troch implementácií MOM: Sun Java Message Queue, Microsoft Message Queuing (ďalej len MSMQ) a MQSeries od IBM. Obsahuje porovnávaciu tabuľku týchto troch produktov, kde je vidno, že každý z nich poskytuje určité vlastnosti: podpora broadcast/multicast, prioritizácia správ, bezpečnosť, garantované doručovanie správ, transakčné spracovanie. V dokumente bol citovaný výkonnostný test MSMQ a MQSeries zverejnený na webovej stránke spoločnosti Microsoft. Výsledky tohto porovnania síce pre nás nie sú zaujímavé (potrebujeme porovnať väčšiu množinu atribútov, zaujímajú nás nekomerčné produkty, ktorým sa porovnanie nevenovalo, a zároveň od roku 2001 sa vlastnosti porovnávaných produktov výrazne rozšírili). Porovnanie spomíname preto, lebo nám slúžilo ako inšpirácia. Ďalšie detaily ohľadom tohto projektu môžete dočítať v [Nat01].

Ďalšie zverejnené dokumenty sa týkali najmä porovnania vzhľadom na výkon jednotlivých produktov. Bolo nájdených viacero porovnaní komerčných poskytovateľov JMS, a len jedno relevantné porovnanie nekomerčných produktov, ktoré obsahovalo pokus o hodnotenie aj iných ako výkonnostných atribútov. V nasledujúcom texte niektoré z týchto porovnaní bližšie popíšeme.

4.1 Sun Java System Message Queue vs. IBM WebSphere MQ

V roku 2003 bol zverejnený porovnávací test výkonnosti medzi produktmi Sun Java System Message Queue 3.5 (predtým Sun ONE Message Queue) a IBM WebSphere MQ 5.3 (predtým IBM MQSeries). Toto porovnenie vykonala spoločnosť Crimson Consulting Group s podporou firmy Sun[8].

Okrem testov dvoch domén (point-to-point, publish/subscribe) a správ veľkosti 10 kilobajtov boli testované perzistentné aj neperzistenté správy a potvrdzovanie správ na strane konzumenta, a to metódou AUTO a DUPS_OK, navyše aj použitie selektorov. Spolu sa používalo 25 testovacích scenárov.

Sun Java System Message Queue bol v doméne point-to-point v porovnaní s IBM WebSphere MQ rýchlejší v priemere 10,5x (maximálne o 2050%), v doméne publish/subscribe v priemere 6.7x rýchlejšie (maximálne o 2080%). Viac o tomto porovnaní môže čitateľ nájsť v [Cri03].

[8] Porovnania, ktorých autorom je jeden z účastníkov porovnania, prípadne ktoré si jeden z účastníkov objedná, považujeme z pochopiteľných dôvodov za neobjektívne. V práci ich uvádzame najmä preto, že sme sa inšpirovali použitými metódami porovnávania.

4.2 Test výkonnosti medzi SonicMQ® a TIBCO Enterprise™

V novembri 2003 bolo zverejnené porovnanie poskytovateľov JMS SonicMQ 5.0.2 a TIBCO Enterprise for JMS 3.1. Testy vykonala a výsledky zverejnila spoločnosť Sonic.

SonicMQ bol rýchlejší v testoch o od 15% do 329%. Ďalšie podrobnosti môžete dočítať v [Son03].

Spoločnosť Tibco spochybnila objektivitu výkonnostných testov (viď. napríklad [Inf03]). Táto situácia nie je výnimočná – testy realizované jednou zo zúčastnených spoločností možno len ťažko považovať za objektívne.

4.3 Nezávislé porovnanie výkonnosti komerčných produktov JMS

Najrelevantnejší nájdený test výkonnosti vedúcich komerčných produktov JMS vykonala spoločnosť Krissoft Solutions v októbri 2004. Porovnávané boli nasledujúce produkty: FioranoMQ 7.5, SonicMQ 6.0, Tibco EMS 4.0, IBM WebSphereMQ 5.3 testovacím programom TestHarness od Sonic, vid. [Kri04].

Bola testovaná doména publish/subscribe, kde konzumenti používali potvrdzovanie správ typu DUPS_OK a producenti posielali správy veľkosti 1KB. Testovanie bolo vykonané v dvoch najpoužívanejších modeloch posielania správ:

- producent poslal neperzistentné správy a netrvalo pripojený konzument ich zobral z témy (NP/NT)
- producent poslal perzistentné správy a trvalo pripojený konzument ich zobral z témy (P/T)

V každom modeli ďalej bol vykonaný test škálovateľnosti servera a škálovateľnosti témy tak, že v prvom prípade bol pre každú tému spustený jeden producent a jeden konzument správ a v druhom prípade bolo pre jednu tému spustených viacero producentov a konzumentov správ. Tieto počty boli postupne nastavované na hodnoty 1, 10, 50.

Výsledky sú zhrnuté do tabuľky, kde je znázornený pomer výkonnosti najvýkonnejšieho produktu (FioranoMQ) voči ostatným - napríklad v škálovateľnosti servera v modeli NP/NT FioranoMQ prekonal SonicMQ od 2 do 4-krát.

pomer výkonnosti FioranoMQ k ostatným produktom		SonicMQ	Tibco EMS	IBM WebSphereMQ
model NP/NT	škálovateľnosť servera	2x-4x	2x-2.5x	13.5x-18,5x
	škálovateľnosť témy	2x-10x	1x-2x	10x-18x
model P/T	škálovateľnosť servera	1x-2x	1x	5x-13x
	škálovateľnosť témy	2x-4x	2x-3x	5x-28x

Tabuľka 4–1: Výsledky nezávislého porovnania výkonnosti komerčných produktov JMS

Ďalšie podrobnosti môžete nájsť v [Kri04].

4.4 Porovnanie štyroch nekomerčných produktov JMS a SonicMQ

Jediný nájdený test výkonnosti porovnajúci aj nekomerčné produkty JMS vykonala spoločnosť LogicBlaze (výrobca ActiveMQ[9]), ktorá výsledky zverejnila v júni 2005. Porovnala nasledujúce produkty: ActiveMQ 3.1, JBossMQ 4.0.1, SonicMQ 6.1 od Progress Software, JORAM RC2, a MantaRay 7.1.

Boli testované štyri spôsoby doručovania správ:

- producent poslal perzistentné správy a trvalo pripojený konzument ich zobral z témy,
- producent poslal perzistentné správy a netrvalo pripojený konzument ich zobral z témy,
- producent poslal neperzistentné správy a trvalo pripojený konzument ich zobral z témy,
- producent poslal neperzistentné správy a netrvalo pripojený konzument ich zobral z témy.

Ďalej, každý spôsob bol vykonaný v troch scenároch:

- jeden producent, jeden konzument a jedna téma,
- desať producentov, desať konzumentov a jedna téma,
- desať producentov, desať konzumentov a desať tém.

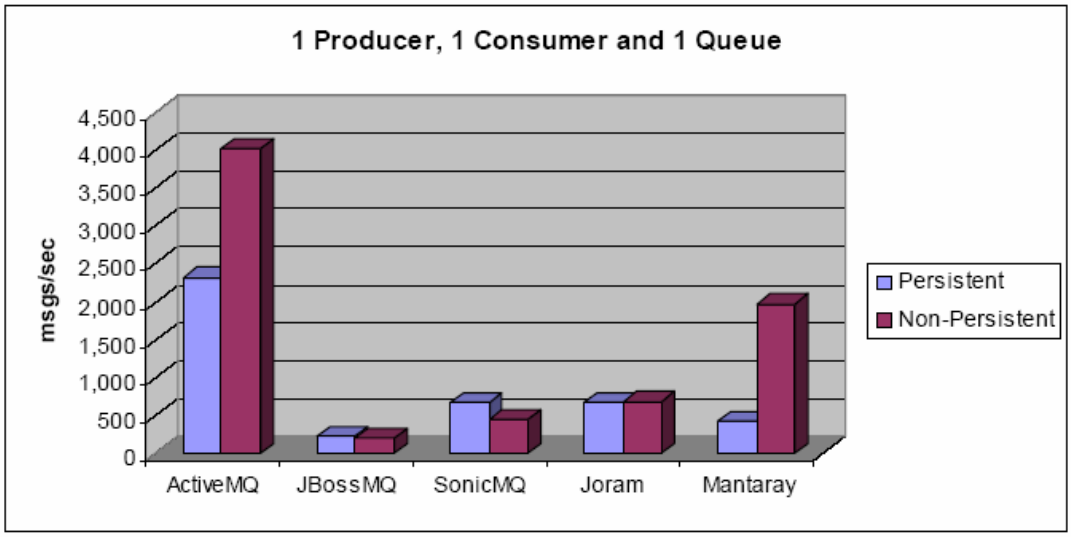
Po analýze údajov z 12 testov môžeme na základe priemerného umiestnenia vyhodnotiť poradie produktov nasledovne: ActiveMQ, SonicMQ, JORAM, JBossMQ. Výsledky testov nájdete v [Log05].

Vzhľadom na to, že na UK v súčasnosti nepredpokladáme použitie modelu publish/subscribe, nebudeme sa týmito výsledkami ďalej zaoberať a sústredíme sa na výsledky dosiahnuté pri testovaní modelu point-to-point.

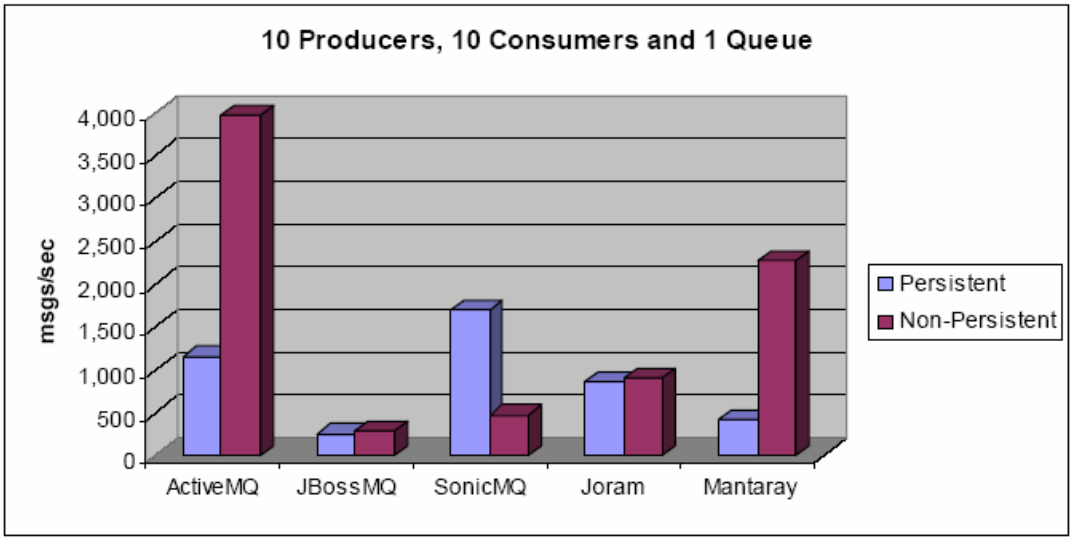
Pri modeli point-to-point boli testované dva spôsoby doručovania správ: producent posielal do radu perzistentné a neperzistentné správy. Každý spôsob bol vykonaný v troch scenároch:

- jeden producent, jeden konzument a jeden rad (Obrázok 4-1),
- desať producentov, desať konzumentov a jeden rad (Obrázok 4-2),
- desať producentov, desať konzumentov a desať radov (Obrázok 4-3).

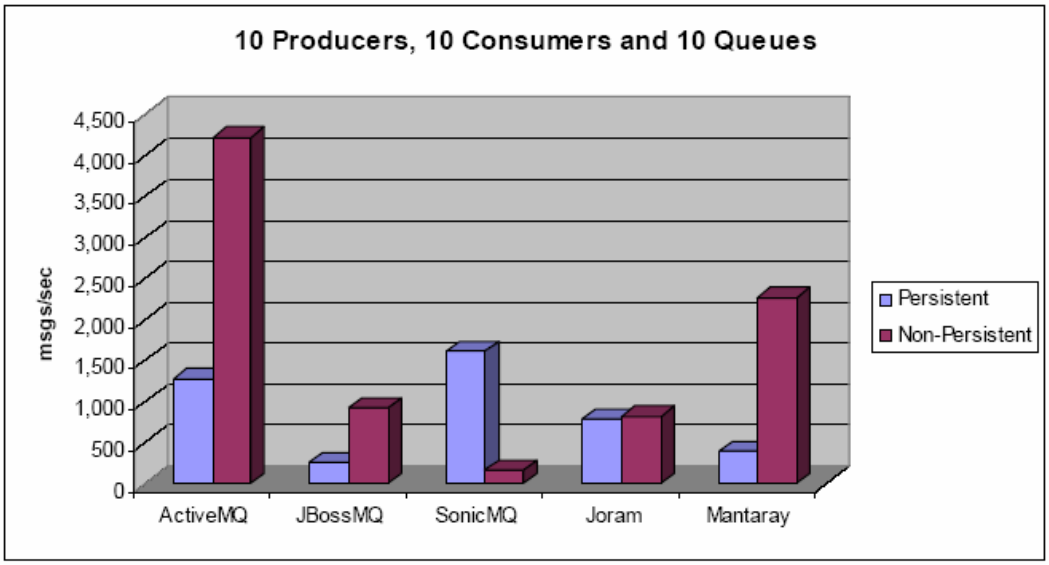
[9] Vzhľadom na to, že ActiveMQ je zároveň jedným z porovnávaných produktov, je dôveryhodnosť tohto testu opäť pomerne nízka.



Obrázok 4-1: Porovnanie výkonnosti nekomerčných produktov (jeden producent, jeden konzument a jeden rad)



Obrázok 4-2: Porovnanie výkonnosti nekomerčných produktov (desať producentov, desať konzumentov a jeden rad)



Obrázok 4-3: Porovnanie výkonnosti nekomerčných produktov (desať producentov, desať konzumentov a desať radov)

Zaujímavé sú výsledky testovania produktu JORAM, kde sú výsledky v prípade perzistentného aj neperzistentného doručovania veľmi podobné – jedným z možných vysvetlení je, že správy sú v rámci tohto produktu spracovávané rovnako bez ohľadu na režim doručovania. V prípade SonicMQ sú výsledky pravdepodobne omylom vymenené, nakoľko spracovanie správ v prípade neperzistentného doručovania by zo zrejmých dôvodov malo trvať kratšie než spracovanie správ so zachovaním perzistencie.

Pre potreby UK je najviac relevantný spôsob doručenia perzistentných správ so scenárom desať producentov, desať konzumentov a jeden rad. Tu je poradie nasledovné: ActiveMQ, JORAM, SonicMQ, MantaRay, JBossMQ (Obrázok 4-2). Viac informácií nájdete v [Log05].

Pokúsili sme sa zistiť podrobnosti ohľadom spôsobu testovania (najmä bodového hodnotenia vlastností produktov, ktorý je súčasťou spomínaného porovnania), ale spoločnosť LogicBlaze nám neposkytla žiadne informácie. Z tohto dôvodu sa týmto porovnaním ďalej nebudeme zaoberať. Poznamenajme, že pri overovaní webových odkazov v apríli 2006 bolo zistené, že zdroj je momentálne nedostupný.

Po hľadaní a skúmaní existujúcich porovnaní poskytovateľov JMS sa pokúsme formou vlastného hodnotenia vybrať produkt čo najlepší pre potreby UK. Toto hodnotenie je popísané v nasledujúcich dvoch kapitolách.

5 Výber produktu

Vzhľadom na to, že nebolo nájdené dostatočne podrobné porovnanie existujúcich poskytovateľov JMS, aby sme vedeli vybrať produkt podľa požiadaviek pre potreby UK, rozhodli sme sa jednotlivé produkty preskúmať a porovnať v rámci tejto diplomovej práce. Postup výberu bol nasledovný:

- najprv bolo vybratých šesť dominujúcich nekomerčných produktov,
- potom bola (na základe dokumentácie k produktom) vytvorená porovnávacia tabuľka produktov z pohľadu vlastností zodpovedajúcich jednotlivým požiadavkám popísaným v kapitole 3. Požiadavky na poskytovateľa JMS,
- v poslednom kroku boli vybrané vlastnosti produktov vyskúšané testovaním.

5.1 Výber dominujúcich nekomerčných produktov JMS

Počas písania diplomovej práce bolo nájdených až 12 nekomerčných produktov JMS. Okrem výhod nekomerčnosti (zadarmo, veľakrát otvorený kód – možnosť úpravy, širšia obec vývojárov atď.) má druh produktov aj niektoré nevýhody, ktorými sa treba zaoberať. Vzhľadom na to, že predpokladáme, že vybratý produkt JMS bude intenzívne používaný na UK minimálne niekoľko rokov, je potrebné sa ubezpečiť hlavne o ďalšom vývoji a o možnostiach podpory. Z týchto dôvodov boli v prvom rade vybraté tie produkty JMS, s ktorými sa na internete stretneme na viacerých miestach okrem vlastnej webovej stránky. Takto sa dá aspoň čiastočne zabezpečiť kvalitu týchto atribútov za predpokladu, že keďže sa o nich „hovorí“, sú pravdepodobne využívané, čo do istej miery zabezpečuje ich ďalší vývoj. Postup bol nasledovný: v [GDi06] boli nájdené linky zaoberajúce sa problematikou JMS, kde boli vybraté nekomerčné produkty JMS, následne rozšírené o ďalšie. Postupne boli vyradené nasledujúce produkty:

- NFC Chat[10] – posledná verzia 1.0.7 bola zverejnená 9.2.2002,
- Mom4j[11] – Google našiel len 286 odkazov na slovo mom4j a posledná verzia 1.1 bola zverejnená 1.12.2004,
- LJMS[12] – projekt bol zrušený na sourceforge.net,
- cyJNDI-cyJMS[13] – Google našiel len 657 odkazov na slovo cyJMS a webová stránka nebola aktualizovaná od roku 2000,
- elemenope[14] – Google síce našiel 9150 odkazov na slovo elemenope, ale počas písania diplomovej práce k tomuto produktu neexistovala dokumentácia[15],

[10] <http://nfcchat.sourceforge.net>
[11] <http://mom4j.sourceforge.net>
[12] <http://sourceforge.net/projects/ljms>
[13] <http://members.aol.com/CAME2BRINGDAPAIN/cyOverview.html>
[14] <http://www.elemenope.org>

- UberMQ[16] – Google našiel len 601 odkazov na slovo ubermq a posledná verzia 2.6 bola zverejnená 29.1.2004[17].

Produkty, ktorými sa budeme zaoberať podrobnejšie v nasledujúcej podkapitole sú:

- Sun Java System Message Queue
- JORAM (Java Open Reliable Asynchronous Messaging)
- ActiveMQ
- OpenJMS
- MantaRay
- JBossMQ

5.2 Porovnanie vlastností vybratých produktov JMS

V tejto podkapitole budú popísané vybraté produkty JMS, ich vlastnosti, porovnáme ich a usúdime, ktoré z nich by boli schopné spĺňať naše požiadavky.

Postup bol nasledovný: po oboznámení sa s dokumentáciou jednotlivých produktov sme vybrali vlastnosti, ktoré sú spoločné pre viaceré produkty a súvisia s požiadavkami sformulovanými v kapitole 3. Požiadavky na poskytovateľa JMS.

Vlastnosti jednotlivých produktov sú zhrnuté v nasledujúcej tabuľke.

[15] viď.: <http://elemenope.org/component/option,com_wrapper/Itemid,41>
[16] <http://sourceforge.net/projects/ubermq>
[17] v rámci aktualizácie odkazov pri poslednej redakcii diplomovej práce bola nájdená novšia verzia 2.7, zverejnená 10.1.2005

číslo požiadavky	produkty JMS:	Sun Java System Message Queue	JORAM	ActiveMQ	OpenJMS	MantaRay	JBossMQ	JBoss Messaging
	vývoj zabezpečuje firma	Sun Microsystems, Inc.	ObjectWeb Consortium	The Codehaus	OpenJMS Group	Coridan, Inc.	JBoss, Inc.	JBoss, Inc.
#1	**podpora synchrónneho a asynchrónneho spracovania**	Áno	Áno	Áno	Áno	Áno	Áno	Áno
#2	**podpora typov správ BytesMessage a TextMessage**	Áno	Áno	Áno	Áno	Áno	Áno	Áno
#3a	**Point-to-Point model posielania správ**	Áno	Áno	Áno	Áno	Áno	Áno	Áno
#3b	**Publish/Subscribe model posielania správ**	Áno	Áno	Áno	Áno	Áno	Áno	Áno
#4	**distribuované transakcie (XA)**	Áno	Áno	Áno	Nie	Áno	Áno	Áno 1.2
#5,#6	**komunikačný protokol medzi klientom a serverom**	TCP, SSL, HTTP, HTTPS, SOAP	Local, TCP, SSL, SOAP	in-VM, TCP, SSL, HTTP, NIO, UDP, multicast, JGroups, JXTA	TCP, SSL, RMI, HTTP,	RMI, HTTP/S, SOAP, TCP, SSL, JCA	TCP, SSL, RMI, HTTP, HTTPS, in-VM	Jgroups 1.2
#5a	**kompresia tela správ na strane klienta**	Áno	Nie	Áno	Nie	Nie	Nie	???
#7	**hardvérové požiadavky**	x86, 256MB RAM, 152MB HDD	x86, 256MB RAM, 34MB HDD	x86, 33MB HDD	x86, 29MB HDD	x86, 3MB HDD	x86, 29 MB HDD	400 MHz CPU, 50MB HDD, 512MB RAM
#8	**implementácia JMS testovaná na platformách (*1)**	Win2k, WinXP, Win2003, Solaris, Linux	Win2000, WinXP, Linux	WinXP SP2, Win2000, Unix	Windows, Linux	WinXP SP1, Win2000, Solaris, Linux	Windows, Linux	???

číslo požiadavky	produkty JMS:	Sun Java System Message Queue	JORAM	ActiveMQ	OpenJMS	MantaRay	JBossMQ	JBoss Messaging
#9	**popularnosť (*2)**	2.150.000	2.740.000	586.000	176.100	306.000	353.000	466.100
	administratívne a monitorovacie nástroje	Áno	Áno	Áno	Áno	Áno	Áno	
	vzdialená administrácia	Áno	Áno	Áno	Áno	Áno	Áno	???
	zabezpečená vzdialená administrácia	Áno	Nie	Nie	Nie	Áno	Áno	???
	zastavenie/obnova prijímania a posielania správ (*3)	Áno	Nie	Nie	Nie	Áno	Áno	???
	monitorovanie zoznamu cieľov (*4)	Áno	Áno	Áno	Áno	Áno	Áno	???
	grafické prostredie pre administráciu a monitorovanie	Áno	Áno	Áno	Áno	Áno	Áno (http)	???
	JMX manažment (*5)	Nie	Áno	Áno	Nie	Áno	Áno	???
#10	**vlastné hodnotenie dokumentácie (*6)**	1	2	2	3	2	2	3
#12	**vysoká dostupnosť - klastrovanie (*7)**	Áno	Áno	Áno	Nie	Nie	Áno	Áno 1.2
#13	**autentifikácia klienta**	Áno	Áno	Áno	Áno	Áno	Áno	Áno
#14	**autorizácia klienta**	Áno	Áno	Áno	Nie	Áno	Áno	Áno
#15	**zabezpečené pripojenie medzi klientom a serverom**	SSL, HTTPS	SSL	SSL	SSL, HTTPS	SSL, HTTPS	SSL, HTTPS	???
#17	**podporované programovacie jazyky pre klienta okrem Javy (*8)**	C, C++		C, C#, Ruby, Perl, Python, Pike, PHP		C++, C#		???

číslo požiadavky	produkty JMS:	Sun Java System Message Queue	JORAM	ActiveMQ	OpenJMS	MantaRay	JBossMQ	JBoss Messaging
#18	**implementovaná verzia špecifikácie JMS**	1.1	1.1	1.1	1.1	1.1	1.1	1.1
#19	**prvá nájdená a uvoľnená verzia:**	Java Message Queue Version 1.1	JORAM first open source version	1.0	v0.2.1	0.9 beta	1.0.0	1.0 Alpha 1
	dátum prvej uvoľnenej verzie:	24.4.2000	26.5.2000	19.8.2004	23.6.2000	29.6.2004	22.8.2001	6.7.2005
	posledná uvoľnená verzia:	3.7 Enterprise Edition	4.3.14	4.0 RC2	0.7.7-alpha-3	1.9	JBossAS 4.0.3SP1	1.0.0 General Availability Release
	dátum poslednej uvoľnenej verzie:	5.12.2005	14.3.2006	3.4.2006	26.12.2005	19.1.2006	30.10.2005	31.3.2006
	licencia	voľne použiteľná	LGPL	Apache 2.0 License	otvorený kód	GPL	LGPL	LGPL
	voľne sťahovateľné z internetu	po registrácií	Áno	Áno	Áno	Áno	Áno	Áno
#20	**spôsob ukladania perzistentných správ (*9)**	Súbor, JDBC	Súbor	JDBC	JDBC	Súbor, JDBC	Súbor, JDBC	JDBC

Tabuľka 5–1: Porovnanie vlastností vybratých nekomerčných produktov JMS

Vysvetlíme popis niektorých skúmaných vlastností, s ktorými sme sa doteraz nestretli v podkapitole 2.6. Java Message Service ani inde v diplomovej práci a na prvý pohľad nie sú zrejmé:

- implementácia JMS testovaná na platformách (*1) – keďže každá implementácia JMS podporuje Javu (a z toho vyplýva, že teoreticky by mala bežať na všetkých platformách, kde existuje JRE implementácia), sú tu naznačené len tie, kde bola táto skutočnosť vyskúšaná,
- populárnosť (*2) – počet nájdených webových odkazov 29. apríla 2006, ktoré obsahujú meno produktu JMS podľa vyhľadávača Google pri aktualizácii porovnávacej tabuľky (len „informatívne kritérium“),
- zastavenie/obnova prijímania a posielania správ (*3) – na strane servera sa v prípade potreby dá zastaviť príjem a/alebo posielanie správ pre konkrétny rad, téma, ktoré nám môže pomôcť pri riešení výnimočných situácií,
- monitorovanie zoznamu cieľov (*4) – funkcionalita, ktorá zabezpečuje monitorovanie zoznamu cieľov (meno cieľa, počet správ v rade/téme, počet konzumentov a producentov, ...),
- JMX manažment (*5) – manažovanie implementácie JMS je zabezpečené cez štandardizované rozhranie, nazvané JMX (Java Management Extensions). Java Developer Kit (JDK) 1.5 napríklad obsahuje grafické prostredie pre manažment prostredníctvom JMX s názvom „Monitoring & Management Console“ (JConsole),
- vlastné hodnotenie dokumentácie (*6) – po študovaní dokumentácie boli porovnané a bodované podľa kvality nasledovne:

  1 - podrobná dokumentácia

  2 - dokumentácia má slabé miesta

  3 - slabá dokumentácia

  4 - žiadna alebo nepoužiteľná dokumentácia,
- vysoká dostupnosť - klastrovanie (*7) – zvýši dostupnosť služby JMS tak, že môžeme pripojiť viacero brokerov[18] do jedného klastra, a tak zabezpečiť vyšší výkon a dostupnosť. Pre klientov je transparentné, cez ktorý server aktuálne komunikuje, a keď server zlyhal, snaží sa klient automaticky pripojiť na ďalší dostupný server bez nutnosti zásahu,
- podporované programovacie jazyky pre klienta okrem Javy (*8) – existencia mapovania rozhrania pre ďalšie programovacie jazyky zabezpečuje možnosť programovať klientov JMS aj v inom programovacom jazyku ako je Java,

[18] inštancie poskytovateľa JMS

- spôsob ukladania perzistentných správ (*9) – Súbor = perzistentné správy sú uložené na lokálnom disku, JDBC = správy sú uložené do databázy podporujúce JDBC rozhranie.

Po popise skúmaných vlastností sa pozrieme na jednotlivé produkty JMS.

5.2.1 Sun Java System Message Queue

Sun Java System Message Queue od Sun vydáva v dvoch verziách, Platform Edition a Enterprise Edition. Pri písaní diplomovej práce bola verzia Platform Edition voľne použiteľná s určitými obmedzeniami týkajúcimi sa funkcionality v porovnaní s platenou verziou Enterprise Edition (klastrovanie, http, https, maximálny počet naraz aktívnych konzumentov na jednom rade bol 2). Pri dokončovaní diplomovej práce však bolo zistené, že od roku 2006 je aj Enterprise Edition voľne použiteľná, preto boli aktualizované údaje v porovnávacej tabuľke, ale už vykonané testy neboli opakované (pridané vlastnosti by nemali mať vplyv na výsledok). Okrem JMX manažmentu podporuje všetky skúmané vlastnosti.

Ďalšie podrobnosti aj o skúmaných vlastnostiach môžete dočítať v [SJSMQ], [SMQEE] a v [MQD05].

5.2.2 Java™ Open Reliable Asynchronous Messaging (JORAM)

JORAM (Java™ Open Reliable Asynchronous Messaging) od ObjectWeb Consortium je produkt s otvoreným kódom pod licenciou LGPL. Je plánované rozšírenie týkajúce sa ukladania správ aj do databázy a podpora C/C++ klientov. JORAM nepodporuje kompresiu tela správ na strane klienta a má menšie nedostatky v oblasti administrácie (zabezpečená vzdialená administrácia, zastavenie/obnova prijímania a posielania správ). Ďalšie podrobnosti môžete nájsť v [JORAM].

5.2.3 ActiveMQ

ActiveMQ od The Codehaus je pomerne nový a aktívne sa vyvíjajúci produkt JMS, od verzie 4.0 je už súčasťou Apache, preto bola aj zmenená licencia na Apache 2.0 License. Administrácia a monitorovanie je podporovaná len priamo cez JMX manažment, t.j. neobsahuje vlastné nástroje. Je možné použiť aj Hermes JMS (viď. [Hermes]) ako grafické užívateľské rozhranie na prácu s radmi. Nie je zabezpečené: zabezpečená vzdialená administrácia, zastavenie/obnova prijímania a posielania správ. Podporuje však ako jediný veľa programovacích jazykov pre klienta a mnoho komunikačných protokolov. Ďalšie detaily môžete dočítať v [Codeh].

5.2.4 OpenJMS

OpenJMS od OpenJMS Group je relatívne pomaly sa vyvíjajúci produkt JMS vyvinutý s otvoreným kódom, ktorý má voči ostatným produktom málo podporovaných vlastností. Ako jediný produkt nepodporuje autorizáciu, distribuované transakcie a nemá dostatočne podrobnú dokumentáciu, čo predstavuje významné nedostatky vzhľadom na požiadavky UK. Navyše, nepodporuje kompresiu tela správ na strane klienta, zabezpečenú vzdialenú administráciu, zastavenie/obnova prijímania a posielania správ, JMX manažment a vysokú dostupnosť – klastrovanie. Podporuje

však automatickú detekciu odpojenia klienta a klienta JMS vo vnútri appletu. Podrobnosti môžete dočítať v [OpenJ].

Najdôležitejšia vlastnosť, ktorú nepodporuje OpenJMS je autorizácia klienta a aj vo vývoji je vidno, že ide pravdepodobne o produkt vyvinutý vo voľnom čase programátorov, ktorí nemajú čas na napísanie podrobnej dokumentácie. Z týchto dôvodov je OpenJMS pre potreby UK považovaný za nevhodný.

5.2.5 MantaRay

MantaRay od Coridan Inc. je vzhľadom k ostatným pomerne nový produkt. Nepodporuje kompresiu tela správ na strane klienta, tiež priamo nepodporuje vysokú dostupnosť – klastrovanie. MantaRay je jediný produkt využijúci plne distribuovanú peer-to-peer technológiu. Z tohto vyplýva, že neexistuje centrálna jednotka (server), ktorú by bolo možné klastrovať, ale z peer-to-peer technológie vyplýva určitá vysoká dostupnosť. Princíp je nasledujúci: keď zlyhá napríklad počítačová sieť medzi dvomi klientskymi inštanciami (peers) MantaRay, tieto budú ukladať správy len na lokálny disk počítača, kde bežia (t.j. offline) a pričom ostatní klienti môžu navzájom komunikovať naďalej (t.j. online). Keď sa sieť zotaví, spojenie medzi príslušnými klientmi sa automaticky nadviaže a dokončí sa posielanie a prijímanie správ. MantaRay Management Console cez webovské rozhranie zabezpečí manažment a monitorovanie všetkých peer-ov na jednom mieste. Podrobnosti môžete dočítať v [Corid].

Vlastnosť peer-to-peer v súčasnosti na UK nie je možné naplno využiť vzhľadom na centralizovanú komunikačnú architektúru (CDO je centrálnym prvkom, cez ktorý prechádza všetka komunikácia).

Aj pri nasadení by sme mali viac práce pri nastavení firewallov, lebo pri centralizovanom JMS potrebujeme na každej fakulte nastaviť prechod cez firewall medzi klientskym počítačom a serverom JMS, ale pri peer-to-peer technológii by sme potrebovali nastaviť pre jeden klientsky počítač teoreticky všetky počítače ostatných fakúlt, s ktorými môže komunikovať. T.j. náročnosť nastavenia firewallov by sa takto zvýšila približne desaťnásobne.

Z týchto faktov vyplýva, že MantaRay je síce zaujímavé riešenie, ale pre potreby UK nevhodné, preto sa ním už podrobnejšie nebudeme zaoberať.

5.2.6 JBossMQ

JBossMQ od JBoss Inc je súčasťou aplikačného servera JBoss. Tento produkt vznikol vývojom zo SpyderMQ[19] pod licenciou LGPL. V súčasnej dobe sa aktívne pracuje na inovácii tohto produktu pod názvom JBoss Messaging. Spoločnosť JBoss tvrdí, že po zverejnení finálnej verzie JBoss Messaging bude podporovať JBossMQ ešte dva roky, z čoho vyplýva, že má zmysel sa ním zaoberať. Okrem kompresie tela správ na strane klienta podporuje všetky vlastnosti vymenované v porovnávacej tabuľke. Ďalšie podrobnosti môžete nájsť v [JBoMQ] a v [JBASD].

[19] <http://groups.yahoo.com/group/spyderMQ>

V 31.3.2006 bola zverejnená prvá verzia JBoss Messaging 1.0.0 General Availability Release pod licenciou LGPL, ktorou sa nám už z časových dôvodov nepodarilo podrobnejšie zaoberať. Počas písania diplomovej práce ešte neexistovala k tomuto produktu podrobná dokumentácia, preto bola porovnávacia tabuľka vyplnená len čiastočne (nenájdené vlastnosti v skromnej dokumentácii sú označené ako ???). Boli spomenuté vlastnosti ako napríklad distribuované transakcie a vysoká dostupnosť, ktorá má byť implementovaná vo verzii 1.2.

Implementácia JMS od JBoss bola vyvinutá odznova najmä preto, lebo v staršej verzii (JBossMQ) bol výkon pomerne nízky. Pri JBoss Messaging bol dôraz kladený na túto oblasť, najmä na priepustnosť a na nízke oneskorenie. Produkt je vyvíjaný najmä pre potreby JBoss ESB, ktorý potrebuje rýchle a spoľahlivé posielanie správ ako základ pre svoju činnosť. Podrobnosti môžete dočítať v [JBMes].

5.3 Výsledky porovnania vlastností vybratých produktov JMS

Keď teraz zosumarizujeme, ktoré produkty by splnili stanovené požiadavky, ostávajú tieto: Sun Java System Message Queue, JORAM, ActiveMQ, JBossMQ.

Každý produkt funguje v sieťach TCP/IP (požiadavka s číslom #5, viď riadok „komunikačný protokol medzi klientom a serverom“ v Tabuľka 5–1: Porovnanie vlastností vybratých nekomerčných produktov JMS). Ako pomôcku pri pomalých linkách môžeme využiť kompresiu tela správ na strane klienta (pož. #5a), ktorú podporuje len Sun Java System Message Queue a ActiveMQ.

Prechod cez firewall (pož. #6) cez HTTP–tuneling zabezpečuje každý produkt, ale keďže potrebujeme aj chránenú komunikáciu, najvhodnejším prostriedkom je HTTPS, ktorý podporuje len Sun Java System Message Queue a JBossMQ. Nemôžeme však vylúčiť ani zvyšné produkty, lebo každý podporuje aspoň SSL, ktoré je pravdepodobne najpoužívanejším riešením ochrany proti odpočúvaniu a neoprávnenej modifikácii pri prenose (pož. #15). Autentifikáciu (pož. #13) a autorizáciu (pož. #14) podporujú všetky zvolené produkty.

Každý produkt je schopný pracovať na bežnom počítači (pož. #7) s nami vybratými operačnými systémami (pož. #8).

Požiadavky použiteľnosti (pož. #9-10) ovplyvnili až deväť riadkov v Tabuľka 5–1: Porovnanie vlastností vybratých nekomerčných produktov JMS, kde je vidno, že Sun Java System Message Queue a JBossMQ je dobre manažovateľný aj z diaľky, ale aj ostatné produkty sa dajú pomerne dobre manažovať lokálne.

Pri skúmaní vlastností týkajúcich sa spoľahlivosti sme našli spôsob ukladania perzistentných správ (do súboru a/alebo do databázy cez JDBC), ktorý úzko súvisí s pož. #11. Bolo by vhodné overiť spoľahlivosť ukladania perzistentných správ vo výnimočných situáciách, ktoré vykonáme v kapitole 6. Testovanie spoľahlivosti vybratých produktov JMS. Vysokú dostupnosť (pož. #12), ktorá je vítaná, podporujú všetky zvolené produkty. Podporu JDBC (pož. #20) sme našli pri každom produkte okrem JORAM.

Každý produkt je voľne sťahovateľný z internetu (pož. #19) – Sun Java System Message Queue požaduje registráciu cez internet – a všetky produkty podporujú špecifikáciu JMS verziu 1.1 (pož. #18).

Nezaoberali sme sa zatiaľ výkonnostnými požiadavkami (pož. #21), požiadavkami na využitú kapacitu (pož. #16) a bolo by treba overiť uloženie perzistentných správ na strane servera (pož. #11), ktorým sa budeme zaoberať v nasledujúcej kapitole.

6 Testovanie spoľahlivosti vybratých produktov JMS

Spoľahlivosť počítačového systému znamená, do akej miery sa dá na daný systém spoľahnúť. Indikuje, v akej miere používateľ môže dôverovať systému, že sa bude správať tak, ako je od neho očakávané. Táto vlastnosť je numericky nevyjadriteľná, preto sa bežne používajú slová ako nespoľahlivý, spoľahlivý a vysoko spoľahlivý systém, aby sa vyjadrilo, ako sa na daný systém spoliehame.

Spoľahlivosť má 4 dimenzie:

- **Dostupnosť** (availability) – pravdepodobnosť, že systém bude (v danom momente) schopný poskytovať služby,
- **Spoľahlivosť** (reliability) – pravdepodobnosť, že systém bude vedieť poskytovať služby správnym spôsobom (tak, ako ich používateľ očakáva),
- **Zabezpečenie** (safety) – posúdenie schopnosti systému fungovať tak, aby nespôsoboval škodu ľuďom a svojmu okoliu,
- **Bezpečnosť** (security) – posúdenie schopnosti systému odolávať náhodnému alebo úmyselnému narušeniu.

Spoľahlivosť a dostupnosť sú dva odlišné pojmy. V [Som02] je ilustrovaný rozdiel nasledovným príkladom: Bežný textový editor je vysoko dostupný, ale nie je veľmi spoľahlivý. Nízku spoľahlivosť editora môžeme kompenzovať tým, že prácu budeme častejšie ukladať a budeme vytvárať bezpečnostné kópie. S dostupnosťou (čiže s poskytovanou službou ako napríklad: zmena písma na tučné, číslovaním strán, odstavca, ...) sa zvyčajne nevyskytujú problémy.

Spoľahlivosť a dostupnosť môžeme definovať nasledovne.

- Spoľahlivosť je pravdepodobnosť bezchybnej prevádzky systému v definovanom časovom období, v definovanom prostredí a daným účelom.
- Dostupnosť je pravdepodobnosť, že v danom okamihu bude systém funkčný a schopný poskytovať požadované služby.

Podľa definície je vidno, že pri posudzovaní spoľahlivosti musíme brať do úvahy aj prostredie a účel, na aký systém použijeme. V našom prípade je možné dostať úplne rozličné výsledky pri skúmaní spoľahlivosti poskytovateľov JMS v identických podmienkach pri zmene operačného systému (napríklad zmenou platformy z windows na unix). Takisto by bolo možné dostať iné výsledky, keď namiesto malého množstva veľkých správ by sme posielali veľa menších správ, čiže by sme zmenili účel.

Ďalšie detaily a techniky zvyšovania spoľahlivosti môžete dočítať v [Som02] odkiaľ boli čerpané aj predošlé poznámky.

Počas vyše polročnej prevádzky Sun Java System Message Queue pracoval spoľahlivo, takže s pripojením (posielaním a prijímaním správ) neboli žiadne problémy[20]. Z tohto dôvodu nebolo vykonanie úplného testu spoľahlivosti v čase určenom na vytvorenie diplomovej práce realizovateľné (takýto test by pravdepodobne preukázal, že produkty za bežných okolností pracujú spoľahlivo). Preto bol vykonaný len test spoľahlivosti konkrétnej funkcionality – zachovania perzistencie správ v prípade neočakávanej udalosti. Z kapitoly 1. Popis problému je vidno, že táto vlastnosť je pre nás v súčasnosti najdôležitejšia.

6.1 Test zotavenia perzistentných správ

Počas písania diplomovej práce bol nájdený jediný relevantný test zaoberajúci sa spoľahlivosťou poskytovateľov JMS. Tento test je zameraný okrem bežných testov rýchlosti aj na test zotavenia perzistentých správ dvoch komerčných poskytovateľov JMS (IBM´s MQ Series, TIBCO´s Rendezvous). Týmto dokumentom sme sa inšpirovali pri našom testovaní zotavenia peristentných správ – požiadavka #11 (správy sa za žiadnych okolností nesmú stratiť). Podrobnejšie je k dispozícii v [QoS04].

6.2 Metodika testovania

Naša situácia na UK už bola popísaná v kapitole 1. Popis problému. Pre simuláciu reálnej situácie sme použili doménu point-to-point, čiže jeden rad, do ktorého producenti zapisujú správy a jeden konzument ich vyberá. Boli použité 2 počítače, jeden klientsky počítač na posielanie a príjem správ a druhý počítač ako server JMS, kde boli nasadení jednotliví poskytovatelia JMS. Dva počítače boli prepojené cez samostatný switch s teoretickou rýchlosťou 100Mbit/s.

Ako metriku na testovanie sme použili počet stratených perzistentných správ v dôsledku umelo vyvolaného zlyhania servera, ktorým sme sa snažili napodobniť situácie vyskytujúce sa v reálnej prevádzke. Cieľom bolo totiž zistiť, za akých podmienok a u ktorých poskytovateľov JMS sa stratia perzistentné správy.

Realizovali sme štyri testovacie scenáre:

1. korektné vypnutie poskytovateľa JMS podľa dokumentácie (StopJMS) – sa dá očakávať, že administrátor bude za istých okolností potrebovať napr. zmeniť konfiguračný súbor a pri tejto príležitosti bude musieť poskytovateľa JMS reštartovať,
2. ukončenie procesu Java.exe cez riadenie úloh (StopJava) – ak sa administrátorovi z nejakých dôvodov nepodarí korektne ukončiť poskytovateľa JMS, ukončí ho pravdepodobne týmto spôsobom,
3. reštartovanie operačného systému (RestartOS) – v prípade, že administrátor z nejakých dôvodov potrebuje reštartovať server a zabudne korektne vypnúť poskytovateľa JMS,

[20] Raz sme boli svedkami toho, že posielanie správ fungovalo, avšak namiesto niekoľkých sekúnd trvalo poslanie jednej správy približne 3 minúty. V rámci riešenia problému bolo zistené, že príčinou bolo pravdepodobne zlyhanie na úrovni operačného systému (t.j. nie samotného SunJSMQ).

4. stlačenie tlačidla reset na serveri (ResetHW) – simulácia výpadkov prúdu[21], ktoré sú pomerne časté napr. na FMFI.

Každá poslaná správa bola označená tak, aby ju bolo možné po prijatí jednoznačne identifikovať – aby sme mali takto možnosť identifikovať stratené správy. Sledované boli nasledujúce metriky:

- počet úspešne poslaných správ (O – odoslaných),
- počet správ prijatých pred zlyhaním poskytovateľa JMS (P – prijatých),
- počet správ prijatých po zotavení poskytovateľa JMS (PPZ – prijatých po zotavení),
- počet tých správ, ktoré boli prijaté pred zlyhaním poskytovateľa JMS aj po jeho zotavení (DP – duplicitne prijatých); tieto správy nepočítame medzi PPZ[22].

Tieto metriky indikujú, či došlo k strate správ alebo nie nasledovne[23]:

- O-P-PPZ = 0 znamená, že všetky perzistentné správy boli korektne uložené na serveri, po zotavení sa korektne poslané správy nestratili,
- O-P-PPZ > 0 znamená, že počas zlyhania servera bolo stratených O-P-PPZ správ,
- O-P-PPZ < 0 znamená, že poskytovateľ JMS doručil aj také správy, ktorých poslanie nebolo úspešne dokončené (vysvetlenie je uvedené nižšie).

Posielali sme 2 typy správ, aby bolo možné simulovanie dvoch reálnych využití na UK:

- BytesMessage – obsahoval v tele správy údaje zo súboru s veľkosťou 512kB (veľkosť správy bola určená podľa použitých veľkostí správ poslaných medzi systémom STUDENT a CDO),
- TextMessage – by obsahoval v tele správy údaje z XML súboru veľkosti 512B (táto veľkosť je potrebná na prenos údajov napr. o jednom študentovi).

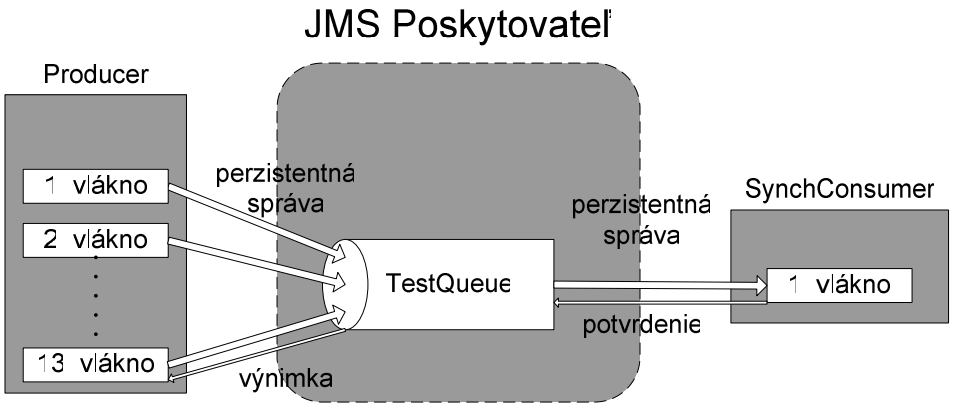
6.2.1 Popis testovacej aplikácie

Testovaciu aplikáciu môžeme znázorniť nasledovne:

[21] Keďže na testovanie boli použité bežné počítače a chceli sme sa vyhnúť vzniku hardvérových porúch bežných pri skutočnom výpadku prúdu, rozhodli sme sa, že túto situáciu budeme simulovať s tlačením tlačidla RESET.

[22] Situácia, kedy je správa prijatá pred zlyhaním aj po ňom, sa môže nastať vtedy, ak poskytovateľ JMS nestihne pred svojím zlyhaním korektne spracovať potvrdenie prijatia správy zo strany konzumenta (potvrdzovanie správ je znázornené na obrázkuObrázok 6-1 aObrázok 6-2). Špecifikácia JMS predpisuje, že takéto nepotvrdené správy majú byť doručené znovu, s nastavením príznaku JMSRedelivered. Dvakrát prijaté správy správne označené ako JMSRedelivered medzi DP nepočítame, avšak tie, ktoré nie sú označené ako JMSRedelivered, áno.

[23] V skutočnosti bolo vyhodnotenie riešené porovnaním zoznamu poslaných a prijatých správ, ako je uvedené nižšie.



Obrázok 6-1: Schéma komunikácie testovacej aplikácie

Producentom bola viacvláknová aplikácia (znázornená ako Producer), kde každé vlákno posiela perzistentné správy poskytovateľovi JMS, až kým nedôjde k simulovanému zlyhaniu servera. Za úspešne poslané správy budeme považovať tie, ktoré boli korektne poslané, t.j. metóda producer.send(...) nesignalizovala výnimku. Správy, pri doručovaní ktorých do TestQueue bola signalizovaná výnimka (t.j. počas ich posielania nastalo zlyhanie servera, kvôli ktorému poskytovateľ JMS nedokázal informovať klienta o uložení správy v rade) nazveme „správy s nedokončeným poslaním“ a ich počet je zistiteľný ako P+PPZ–O[24].

Pri synchrónnom spôsobe prijímania správ (aplikácia SynchReceiver) bola použitá metóda potvrdzovania správ AUTO_ACKNOWLEDGE, ktorá zabezpečuje, že po úspešnom zavolaní metódy consumer.receive() je vykonané aj automatické potvrdzovanie správ. Keď metóda consumer.receive() počas príjmu signalizuje výnimku, to znamená, že správa by mala zostať v rade a byť doručená po zotavení, a to s príznakom JMSRedelivered.

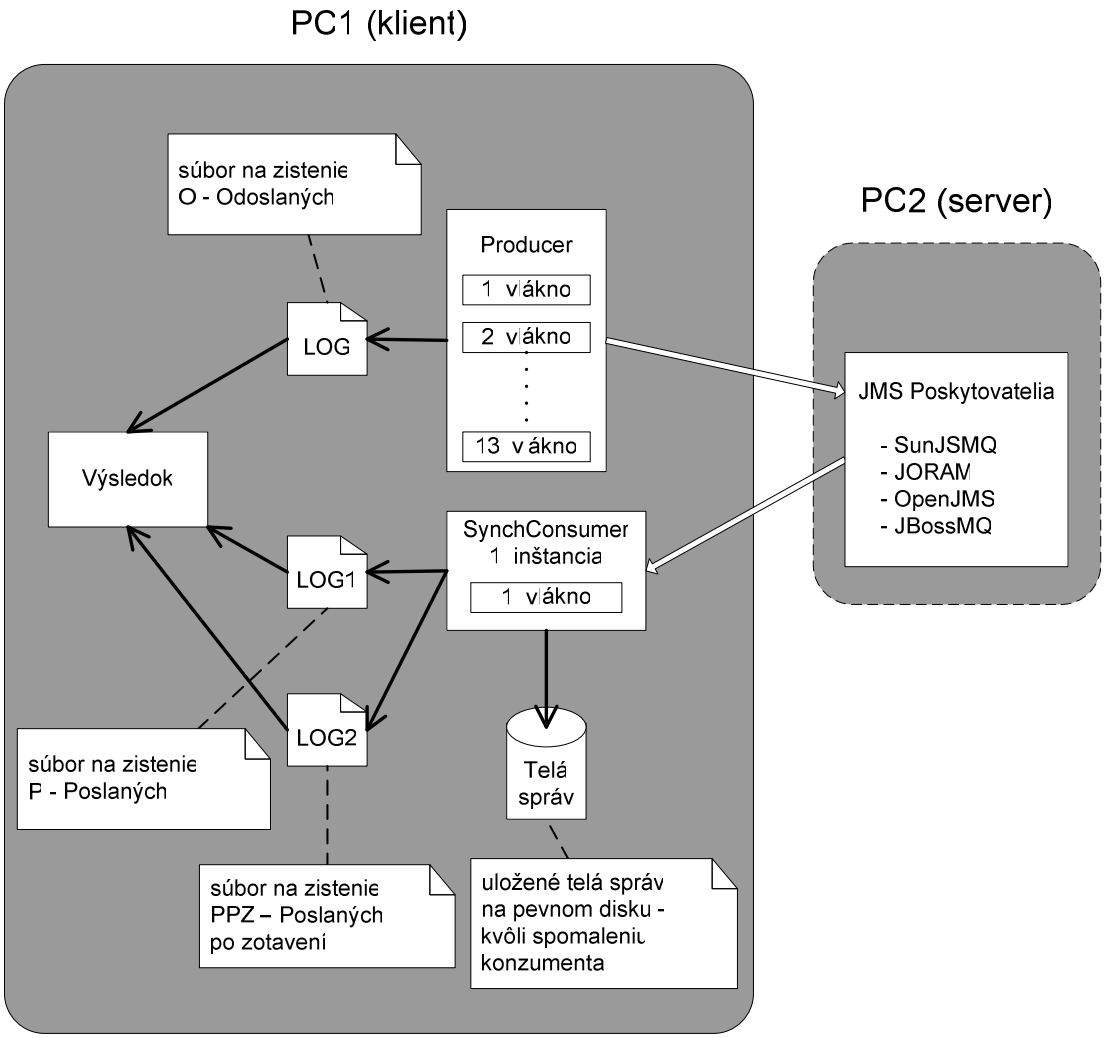
Správy boli vytvorené len raz a telo správy bolo doplnené údajmi z pripravených súborov. Ďalej hlavička správ obsahovala na identifikáciu číslo vlákna producenta a číslo správy. Podľa týchto údajov mal konzument možnosť zaznamenať, ktoré správy boli prijaté a ktoré boli stratené.

Konzument sa v nekonečnom cykle snažil prijať správy z príslušného radu (TestQueue). Po prijatí každej správy vytvoril na základe informácií z jej hlavičky (číslo vlákna producenta a číslo správy) názov súboru, do ktorého uložil telo správ. Príklad názvu súboru je: out\SunJSMQ\StopJMS\instancie_01\msg_0000001.xml, kde vidno všetky podstatné atribúty týkajúce sa testu spoľahlivosti.

Ďalej konzument bol implementovaný tak, aby sledoval, či správu už raz prijal. Vtedy namiesto msg_0000001.xml použil názov exist_msg_0000001.xml, resp. v prípade, že poskytovateľ JMS nastavil vlastnosť JMSRedelivered na pravdivú, tak ako redelivered_msg_0000001.xml. Takto sme boli schopní sledovať aj to, či správne používa nastavenie JMSRedelivered a či nie sú po zotavení správy posielané ešte raz.

[24] Ich počet je –(O-P-PPZ), t.j. P+PPZ-O.

Proces zisťovania výsledkov a komponenty testovacej aplikácie sú uvedené na nasledujúcom obrázku:



Obrázok 6-2: Štruktúra testovacej aplikácie

Producent pri posielaní správ ukladal do logu záznamy obsahujúce číslo vlákna a číslo správy – tento log súbor slúžil na zistenie počtu úspešne poslaných správ (O). Konzument takisto vyrobil log, jeden pred vyvolaním umelej poruchy – na zistenie počtu prijatých správ (P – poslaných) a jeden po zotavení na zistenie PPZ (prijatých po zotavení). Tieto súbory boli analyzované aplikáciou Výsledok, ktorej výstupom sú údaje uvedené v tabuľke Tabuľka 6–2.

Naším cieľom bolo, aby súčasne bežalo trinásť vlákien producenta[25] a jeden konzument tak, aby na strane servera v prípade BytesMessage bolo uložených a čakajúcich na príjem spolu približne 100MB správ a v prípade TextMessage 1MB správ. Keďže od producentov boli posielané správy dopredu pripravené v pamäti

[25] Pri komunikácii Študent-CDO máme trinásť inštancií systému Študent, z ktorých každý posiela údaje do CDO.

a konzument správy ukladal na pevný disk, bolo dosiahnuté dostatočné spomalenie konzumenta voči producentom.

6.2.2 Nasadenie poskytovateľov JMS

Pri nasadení poskytovateľov JMS sme postupovali podľa inštrukcií v dokumentácii poskytovateľov JMS. Mali sme k dispozícii z webu najaktuálnejšiu stabilnú verziu dostupnú počas písania diplomovej práce, následne inštalovanú. Potom bol vytvorený rad s názvom testQueue (Destination – cieľ), a keď distribúcia obsahovala aj službu JNDI (Java Naming and Directory Interface – viď [Jnd99]) tak bol okrem radu vytvorený aj testCF (ConnectionFactory – objekt obsahujúci informácie o adrese a porte, kde sa poskytovateľ JMS nachádza).

Sun Java System Message Queue 3.6 Platform Edition (SunJSMQ)

Táto distribúcia bola bez služby JNDI, preto informácie pre ConnectionFactory boli priamo nastavené v testovacej aplikácii. Na ukladanie perzistentných správ je štandardne použitý súbor, ktorý bol použitý aj v našom prípade.

V dokumentácii je popísaný atribút imq.persist.file.sync.enabled (štandardne vypnutý), ktorý keď je zapnutý, SunJSMQ bude po každom prijatí perzistentnej správy synchronizovať rad s obsahom súboru. Táto synchronizácia spôsobí spomalenie, ale vylučuje stratu perzistentných správ. Z tohto dôvodu bol preskúmaný spomínaný produkt bez zapnutia (SunJSMQ), aj so zapnutím (SunJSMQ-sync) tohto atribútu. Táto zmena bola vykonaná v súbore config.properties.

Java Open Reliable Asynchronous Messaging 4.3.14 (JORAM)

Táto distribúcia bola so službou JNDI, ktorá bola využitá na získanie informácie pre ConnectionFactory (testCF) a Destination (testQueue).

V dokumentácii je popísaný atribút Transaction, ktorý je nastavený štandardne na hodnotu fr.dyade.aaa.util.NullTransaction. Z dokumentácie vyplývalo, že keď máme využívať perzistentné správy, tak treba zmeniť z NullTransaction na NTransaction. Táto zmena bola vykonaná v súbore a3servers.xml[26]. Na ukladanie perzistentných správ bol použitý súbor, ktorý je na tento účel jediným možným riešením.

ActiveMQ

Táto distribúcia bola bez služby JNDI, preto informácie pre ConnectionFactory boli priamo nastavené v testovacej aplikácii. Na ukladanie perzistentných správ štandardne používa databázu Derby od Apache so spojením s vysoko výkonným žurnálom.

Tento poskytovateľ JMS má vážne nedostatky týkajúce sa perzistencie správ. Počas písania diplomovej práce bola aktuálna verzia ActiveMQ 4.0 M4 Release, ktorá sa po simulovanom zlyhaní (konkrétne počas obnovy žurnálu) dostala

[26] Bol vykonaný pokus aj o testovaní s pôvodnými nastaveniami, ale výsledky boli porovnateľné (nie však lepšie, ako pri našom nastavení).

do nekonečného cyklu. Vzhľadom na to, že navyše bolo zistené, že rad v ActiveMQ má obmedzenú kapacitu[27], ktorá je pre potreby UK príliš malá, bolo rozhodnuté tento produkt vynechať z testovania spoľahlivosti. K jeho testovaniu by sa dalo vrátiť neskôr, keď bude k dispozícii verzia, ktorá nebude mať tieto obmedzenia alebo bude možné kapacitu jednotlivých radov meniť.

ActiveMQ poskytuje aj ďalšie možnosti na ukladanie perzistentných správ. Namiesto Apache Derby umožňuje použiť inú databázu, podporujúcu JDBC, dokonca aj replikovaný sklad údajov.

OpenJMS 0.7.6.1 (OpenJMS)

Posledná zverejnená verzia je 0.7.7-alpha-3, vydaná koncom roka 2005. Táto verzia síce spĺňa špecifikáciu JMS 1.1, ale má ohraničenie veľkosti správ na 32KB, čo pre účely UK nevyhovuje. Toto ohraničenie existuje od verzie 0.7.7-alpha-1, aj keď v tejto verzii nie je dokumentované. Posledná zverejnená stabilná verzia bola 0.7.6.1, vydaná 6. mája 2004, ktorá však spĺňa len špecifikáciu JMS 1.0.2. a kvôli tomu nespĺňa pož. #18.

Táto distribúcia obsahovala službu JNDI, ktorá bola využitá na získanie informácie o ConnectionFactory a Destination. Na perzistenciu správ je používané štandardne Apache Derby.

Poznámka: Pred zotavením bolo potrebné odstrániť súbor bin\openjms.lock, ktorý slúži na to, aby nebolo možné naštartovať dvakrát ten istý server JMS. Keďže pri testovaní bol server ukončovaný neštandardným spôsobom, súbor bolo treba odstrániť manuálne.

JBoss Application Server 4.0.3 SP1 (JBossAS)

JBossMQ je súčasťou aplikačného servera JBoss. Použitá bola verzia 4.0.3 SP1. Pri inštalácii bola zvolená len inštalácia JMS 1.1 server profilu, ostatné komponenty aplikačného servera nie.

Táto distribúcia obsahovala službu JNDI, ktorá bola využitá na získanie informácie o ConnectionFactory a Destination. Na ukladanie perzistentných správ JBossMQ využíva Hypersonic databázu (HSQLDB)[28].

6.3 Realizácia testovania

V tomto odseku vykonáme testovacie scenáre popísané v predošlých odsekoch pre vybrané produkty.

6.3.1 Popis testu

Každý test bol vykonaný pri nasledujúcich podmienkach:

[27] Súbor obsahujúci žurnál má veľkosť 2x20 MB, ktorý na ukladanie správ sumárnej veľkosti 200MB nie je postačujúci, vid. požiadavku #16a. Preplniť rad sa nám podarilo posielaním 46 náhodných správ veľkosti 0,5MB.
[28] <http://hsqldb.org>

- poskytovateľ JMS a klientske aplikácie boli spustené na rôznych počítačoch, aby boli vylúčené chyby týkajúce sa zisťovania presného počtu poslaných a prijatých správ. Testovacia aplikácia totiž ukladá čísla poslaných a prijatých správ do logovacieho súboru, ktorý by mohol byť pri scenári „ResetHW servera“ poškodený,
- producenti a konzument bežali na tom istom počítači, aby sme mali možnosť na jednom mieste sledovať informácie o počte poslaných a prijatých správ,
- pred spustením testovania bol vyprázdnený rad,
- ako prvý bol spustený konzument, ktorý po spustení čakal na správy,
- po spustení aplikácie na posielanie správ bolo vytvorených 13 producentov (každý vo vlastnom vlákne), ktorí paralelne posielali správy,
- po 1 minúte od chvíle, keď začali všetci producenti posielať správy (ktoré boli zároveň prijímané konzumentom), bola na strane servera vykonaná umelá porucha podľa jedného z testovacích scenárov (StopJMS, StopJava, RestartOS, ResetHW),
- po vykonaní umelej poruchy na strane servera, bola na klientskej strane ukončená činnosť producentov a konzumenta s tlačením CTRL+C a následne bol testovaný poskytovateľ JMS znova spustený. Toto ukončenie bolo potrebné vykonať, nakoľko niektorí poskytovatelia JMS podporovali automatickú obnovu pripojenia, a preto pri zlyhaní servera klientske aplikácie nesignalizovali výnimku, len čakali, kým nebol poskytovateľ JMS znova pripravený,
- po zotavení poskytovateľa JMS bol znova spustený konzument správ a boli prijaté všetky správy, ktoré na strane servera v testovacom rade zostali,
- pri všetkých vykonaných testoch bola použitá rovnaká hardvérová platforma, operačný systém, ako aj nastavenie počítačovej siete,
- pred testovaním bolo v operačnom systéme vypnuté kešovanie zápisu na pevnom disku servera. Takto sme uistili poskytovateľa JMS o tom, že keď vykonal uloženie perzistentných správ, príslušné údaje boli na pevný disk skutočne zapísané,
- najprv bol každý test vykonaný raz, a po vyhodnotení boli úspešné testy opakované za účelom potvrdenia výsledku. (Neúspešné testy opakovať netreba, nakoľko akúkoľvek stratu správy považujeme za nesplnenie požiadaviek kladených na implementáciu JMS – pož. #11 v kapitole 3. Požiadavky na poskytovateľa JMS).

Testovacia hardvérová konfigurácia:

	server JMS	producenti/konzument
CPU	Intel Celeron 2 GHz	Intel Pentium 4 2,8 GHz
pamäť	512MB RAM	1GB RAM
pevný disk	80GB Maxtor ATA	80GB Maxtor ATA
operačný systém	MS Windows Server 2003 Standard Edition SP1	MS Windows XP Professional SP2

Tabuľka 6–1: Testovacia hardvérová konfigurácia

6.3.2 Očakávaný výsledok

Predpokladáme, že poskytovatelia JMS používajú kvôli zväčšeniu rýchlosti vlastný, vnútorný keš (cache). V tomto keši pravdepodobne uchovávajú správy tak, aby bolo možné ich na pevný disk zapisovať vo väčších dávkach, typicky vtedy, keď sa keš naplní. Pri takýchto poskytovateľoch JMS môže preto prísť v prípade neočakávaného zlyhania k strate správ uložených v keši, avšak ešte nezapísaných na disk.

Pri scenári StopJMS nepredpokladáme stratenie správ. V prípade, že sa poskytovateľ JMS korektne ukončí pri vypnutí operačného systému, nepredpokladáme stratu správ ani pri scenári RestartOS. Pri scenároch StopJava a ResetHW sa môžu objaviť nejaké nedostatky.

Najhorší prípad sa nastane vtedy, keď poskytovateľ JMS správy vôbec neukladá na pevný disk, len do pamäte. Takto môžeme stratiť všetky správy, ktoré neboli pred umelo vyvolanou poruchou prijaté.

6.3.3 Výsledky testov

Výsledky testu sú zhrnuté v nasledujúcej tabuľke:

	BytesMessage (0,5MB)				TextMessage (512B)			
	StopJMS	StopJava	RestartOS	ResetHW	StopJMS	StopJava	RestartOS	ResetHW
SunJSMQ:	úspešný*	úspešný	úspešný*	NEúspešný	úspešný*	úspešný	úspešný	NEúspešný
poslaných (O)	370	520	540	391	30854	31348	35930	36499
prijatých (P)	68	72	72	62	3114	4704	3359	3573
prijatých po zotavení (PPZ)	312	448	474	12	27742	26644	32571	1
znova prijatých (DP)	0	0	0	0	0	0	0	84
O-P-PPZ	-10	0	-6	317	-2	0	0	32925
SunJSMQ-sync:	úspešný*	úspešný	úspešný*	úspešný	úspešný*	úspešný	úspešný*	úspešný
poslaných (O)	409	374	379	377	2115	1439	1662	1882
prijatých (P)	48	46	39	43	134	124	137	142
prijatých po zotavení (PPZ)	371	328	351	334	1992	1315	1536	1740
znova prijatých (DP)	0	0	0	0	0	0	0	0
O-P-PPZ	-10	0	-11	0	-11	0	-11	0
JORAM	NEúspešný	NEúspešný	NEúspešný	úspešný*	NEúspešný	NEúspešný	NEúspešný	NEúspešný
poslaných (O)	209	198	166	221	520	476	715	447
prijatých (P)	23	23	21	26	49	46	65	44
prijatých po zotavení (PPZ)	176	164	139	198	459	417	637	397
znova prijatých (DP)	0	1	1	1	1	1	1	0
O-P-PPZ	10	11	6	-3	12	13	13	6
OpenJMS	úspešný	úspešný*	NEúspešný	NEúspešný	úspešný	úspešný*	NEúspešný	NEúspešný
poslaných (O)	110	274	227	50	120	1128	1306	87
prijatých (P)	12	23	22	8	24	134	124	19
prijatých po zotavení (PPZ)	98	252	0	0	96	995	0	0
znova prijatých (DP)	0	0	0	0	0	0	0	0
O-P-PPZ	0	-1	205	42	0	-1	1182	68
JBossAS	úspešný*	úspešný	úspešný	NEúspešný	úspešný	úspešný	úspešný	NEúspešný
poslaných (O)	111	77	395	123	4110	4110	16034	15964
prijatých (P)	14	9	29	16	182	182	668	885
prijatých po zotavení (PPZ)	103	68	366	92	3928	3928	15366	14581
znova prijatých (DP)	0	0	0	2	0	0	0	27
O-P-PPZ	-6	0	0	15	0	0	0	498

Tabuľka 6–2: Výsledky testu zotavenia perzistentných správ

Neúspešne dopadli tie testovacie scenáre a prislúchajúci poskytovatelia JMS, ktorí počas umelo vytvorenej poruchy neukladali správy korektne, t.j. prijatých bolo menej správ, než bolo odoslaných. Úspešne dopadli tí, ktorí robili presne to, čo bolo očakávané, čiže po zotavení doručili len tie správy, ktoré boli korektne poslané a doteraz ešte neprijaté. Hodnotenie „úspešný*“ dostali tí, ktorí okrem všetkých správ, ktoré boli producentmi poslané korektne, uložili a doručili aj správy s nedokončeným poslaním, t.j. tie, pri posielaní ktorých metóda producer.send(...) signalizovala výnimku (bližšie vysvetlenie je uvedené vyššie).

Z hľadiska korektnosti je najlepšie riešenie „úspešný“. Z hľadiska zachovania poslaných správ je prijateľný aj výsledok „úspešný*“. Výsledky ohodnotené slovom „neúspešný“ počas tohto testu zlyhali.

Pozrime sa teraz na výsledky jednotlivých testov podrobnejšie.

SunJSMQ doručuje aj niektoré zo správ s neukončeným poslaním. Bohužiaľ sa v testoch ResetHW stratilo pri posielaní malých správ viac než 30 000 správ. Pri analýze situácie na strane servera bolo zistené, že súbor slúžiaci na uloženie perzistentných správ bol poškodený. Toto malo za následok nemožnosť rozoznať hranice medzi správami, a teda stratu všetkých správ uložených za miestom poškodenia.

Pri zmene konfigurácie na SunJSMQ-sync všetky testy prebehli úspešne. Pri posielaní krátkych správ sa prejavilo očakávané spomalenie, a to približne 20-násobné. Príznak JMSRedelivered bol použitý vždy správne. V tomto prípade išlo vždy najviac o jednu správu, a to v prípade, že porucha bola vyvolaná po jej prijatí, ale ešte pred potvrdením jej prevzatia. V prípade ostatných poskytovateľov nebolo zaznamenané použitie JMSRedelivered (počas testovania nebola takto označená ani jedna správa). To znamená, že príslušné produkty pravdepodobne túto vlastnosť nepodporujú, aj keď podľa špecifikácie JMS by mali.

Testovanie JORAM ukázalo, že tento produkt neuchováva perzistentné správy ani pri korektnom vypnutí (scenár StopJMS). Chybu v testovacej aplikácii môžeme vylúčiť vďaka tomu, že pri prenose správ bez umelo vyvolanej chyby na serveri sú všetky správy doručené. Všimli sme si, že z každého vlákna je stratená maximálne jedna správa. Tento produkt pri zotavení ukladá každú správu do samostatného súboru, čo má za následok výrazne vyšší čas potrebný na zotavenie (pri 3000 správach cca 3 minúty, pričom v prípade ostatných produktov trvalo zotavenie niekoľko sekúnd).

Zaujímavý je fakt, že výsledok testovacieho scenára „ResetHW“ bol v prípade produktu JORAM (napriek mnohonásobnému opakovaniu) vždy „úspešný*“. Výsledky ostatných testov sú pritom neúspešné. Túto skutočnosť nevieme do uzávierky odovzdania diplomovej práce vysvetliť.

OpenJMS pri scenároch StopJMS a StopJava zachová všetky poslané správy, avšak v ostatných scenároch (RestartOS, ResetHW) nie je po zotavení doručená ani jedna správa. Na strane servera sa pri zotavení objavuje v logu OpenJMS chybová správa „failed to create database“, ktorá je pravdepodobne príčinou nemožnosti obnoviť ani jednu správu.

JBossAS patrí medzi relatívne pomalšie implementácie JMS, čo zodpovedá faktu, že prakticky vo všetkých testovacích scenároch sa správy nestrácajú. Strata nastáva len v prípade scenáru „ResetHW“, podobne, ako je to pri SunJSMQ. Spoločnosť

JBoss uvádza, že zvýšenie spoľahlivosti je možné dosiahnuť zmenou databázy použitej na ukladanie perzistentných správ[29].

6.4 Súhrn

Na výsledkoch je viditeľné, že použitie kešu v mnohom pomáha pri menších správach dosiahnuť vyššiu rýchlosť spracovania na strane servera, ale s rizikom straty väčšieho množstva správ. Dôvodom tohto je, že uloženie správ do pamäte počítača je omnoho rýchlejšie, ako uloženie správ na pevný disk. Čím je správa menšia, tým viac správ môžeme stratiť.

Pri SunJSMQ sa dá základná konfigurácia (t.j. vypnuté nastavenie okamžitej synchronizácie súboru na ukladanie perzistentných správ) považovať za dobre zvolenú, lebo chyby spôsobené výpadkom prúdu, ktoré boli testované ako testovací scenár ResetHW, je možné efektívne eliminovať použitím záložných zdrojov a ostatné testovacie scenáre zvládol úspešne. Naviac vo väčšine prípadov sa dalo zotaviť aj správy s nedokončeným poslaním[30] a tiež je možné zvýšiť spoľahlivosť za cenu zníženia výkonu. Produktom prakticky vyhovujúcim kritériám spoľahlivosti (konkrétne pož. #11) je tiež JBossAS.

Z týchto testov vyplýva, že použitie okamžitej synchronizácie súboru na ukladanie perzistentných správ je „drahé“, a treba si rozmyslieť, či si vyberieme rýchlejšie spracovanie alebo bezpečnejšie riešenie použitím perzistentných správ.

6.5 Výber produktu spĺňajúceho požiadavky na poskytovateľa JMS.

Podľa kapitoly 5. Výber produktu splnili nami stanovené požiadavky pre UK: Sun Java System Message Queue, JORAM, ActiveMQ, JBossMQ.

Pri vykonaní testu spoľahlivosti sme zistili, že JORAM nespĺňa požiadavku #11 z kapitoly 3. Požiadavky na poskytovateľa JMS. ActiveMQ nespĺňa požiadavku #16 (kapacitné požiadavky). Pri JBossMQ boli zistené nedostatky pri skúmaní spoľahlivosti, ktoré je však podľa informácií z webovej stránky dodávateľa možné riešiť výmenou databázy slúžiacej na ukladanie perzistentných správ. Z tohto faktu vyplýva, že pre potreby UK je najvhodnejším produktom Sun Java System Message Queue, pri ktorom je zmenou konfiguračného súboru možné zabezpečiť aj požiadavku #11.

[29] viď. <http://wiki.jboss.org/wiki/Wiki.jsp?page=ConfigJBossMQDB>
[30] správy, ktoré pri zavolaní metódy producer.send(..) signalizovali výnimku

7 Návrh a implementácia adaptéra

Táto kapitola popisuje riešenie problému komunikácie Študent-CDO a CDO-VIKUK popísané v kapitole 1. Popis problému.

7.1 Princíp komunikácie Študent-CDO

Táto podkapitola popisuje riešenie problému komunikácie medzi systémom Študent a CDO. Adaptér zabezpečujúci túto komunikáciu bol nazvaný XCDO.

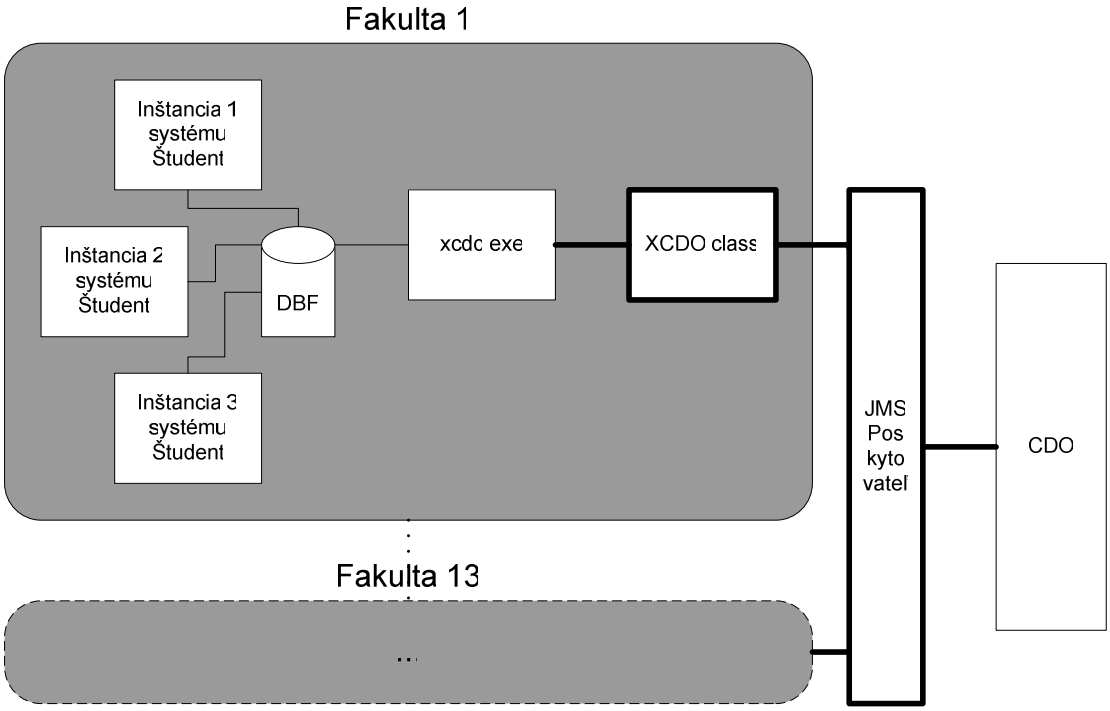
V princípe sú dve možnosti na posielanie údajov do CDO:

1. v systéme Študent bude príslušná funkcia, ktorá exportuje potrebné údaje a zavolá externý adaptér s príslušnými parametrami, ktorý zaistí ich poslanie do CDO („hlavný program“ bude teda systém Študent),
2. alebo adaptér bude bežať v pozadí a automaticky, resp. na pokyn používateľa zavolá príslušnú funkciu systému Študent na vyexportovanie údajov a postará sa o ich doručenie do CDO („hlavný program“ bude teda XCDO).

Najmä vzhľadom na to, že systém Študent je staršia aplikácia vytvorená v programovacom jazyku Clipper, a teda sú možné len jej obmedzené úpravy, zvolili sme druhú možnosť.

Správca systému Študent nám poskytol program (xcdo.exe), ktorý pri zavolaní s príslušnými parametrami zaistí exportovanie potrebných údajov do súboru, resp. importovanie údajov, ktoré prišli z CDO, do systému Študent. Navyše vracia informáciu o úspešnosti alebo neúspešnosti operácie prostredníctvom exit-kódov.

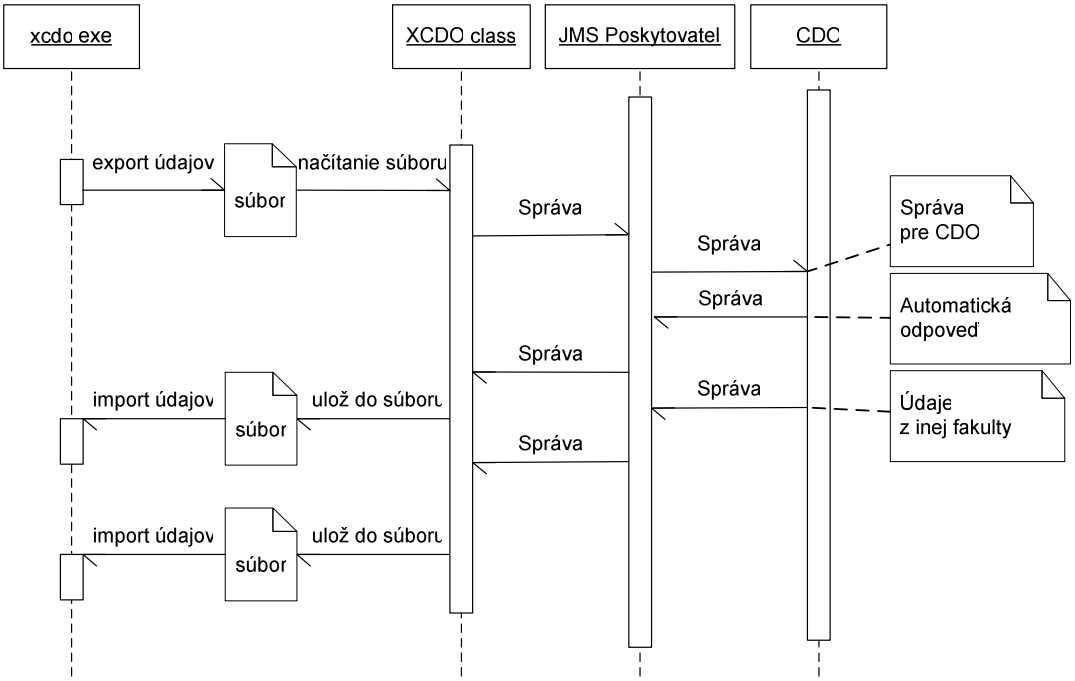
Komunikácia Študent a CDO je znázornená na nasledujúcom obrázku:



Obrázok 7-1: Schéma komunikácie Študent-CDO

Každá fakulta má vlastný server obsahujúci databázu študentov (vo forme súborov DBF), ktorú používa viac inštancií systému Študent na rôznych pracovných staniciach. Správcom poskytnutý program xcdo.exe vie čítať (aj zapisovať) potrebné údaje priamo z databázy. Údaje na pokyn adaptéra XCDO (znázorneného ako XCDO.class) vyexportuje do určeného súboru. Adaptér XCDO získané údaje následne zabalí do správy a pošle poskytovateľovi JMS do príslušného radu. Odtiaľto ich prevezme CDO. Obdobným spôsobom funguje aj spätné doručenie správ a importovanie do systému Študent na príslušnej fakulte. Našou úlohou bolo zabezpečiť hrubou čiarou znázornené komponenty systému.

Komunikácia medzi systémom Študent a CDO je znázornená na nasledujúcom sekvenčnom diagrame:



Obrázok 7-2: Sekvenčný diagram na znázornenie komunikácie Študent – CDO

XCDO na pokyn používateľa (alebo automaticky každý pracovný deň ráno po zapnutí počítača) zavolá externý program xcdo.exe na exportovanie údajov zo systému Študent do súboru, ktorý zabalí do správy a pošle cez zvoleného poskytovateľa JMS do CDO. CDO následne údaje v správe spracuje, vygeneruje odpoveď a pošle cez poskytovateľa JMS. XCDO asynchrónnym spôsobom dostane túto správu, ktorú následne uloží a zavolá externý program na importovanie údajov do systému Študent. Okrem vytvorenia automatickej odpovede môže CDO v rámci spracovávania údajov poslať správy aj iným fakultám.

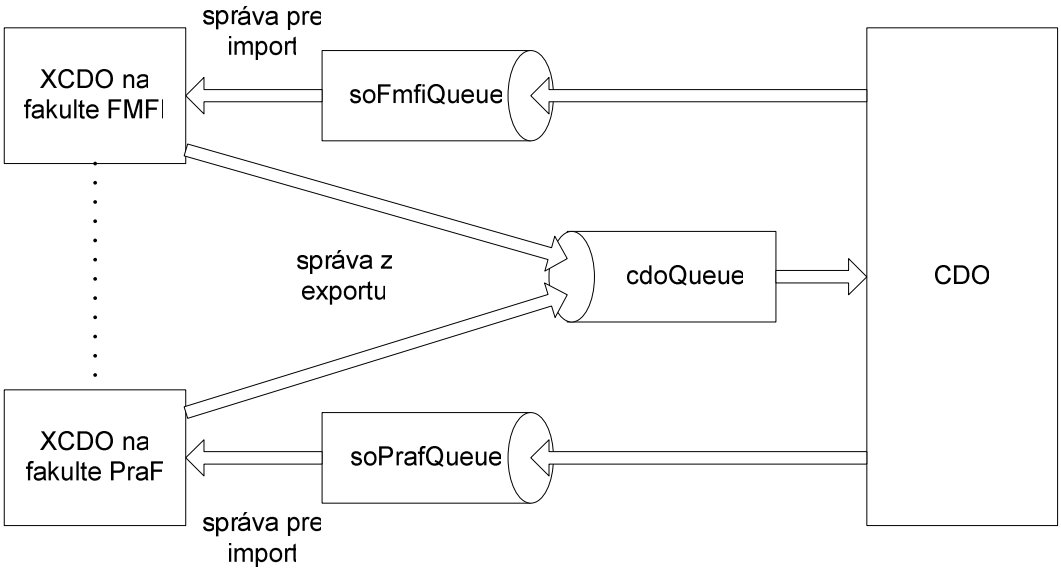
Vzhľadom na to, že na právnickej fakulte je sieť študijného oddelenia izolovaná od univerzitnej siete, je nutné export (a import v opačnom smere) údajov zabezpečiť na druhom počítači a premiestniť vyrobený súbor na počítač obsahujúci XCDO s prístupom do univerzitnej siete (tento modifikovaný spôsob komunikácie budeme nazývať „komunikácia offline“).

Podrobný popis rozhrania medzi XCDO a CDO je uvedený v prílohe A – Popis rozhrania medzi XCDO a CDO.

Topológia prislúchajúcich radov vyzerá nasledovne:

- pre každú fakultu bol vytvorený jeden rad,
- jeden rad bol vytvorený pre CDO,
- XCDO posiela exportované údaje do radu prislúchajúceho CDO,
- CDO posiela údaje pre import do radu prislúchajúceho fakulte.

Grafické znázornenie vyzerá nasledovne:



Obrázok 7-3: Topológia prislúchajúcich radov pre komunikáciu Študent-CDO

Pozrime si, ako bude vyzerať adaptér XCDO z používateľského hľadiska.

7.2 Používateľské hľadisko

Adaptér XCDO je viacvláknová aplikácia na strane fakulty (jedna inštancia pre jednu fakultu), ktorá zabezpečuje nasledovnú funkcionalitu:

F1. exportovanie údajov zo systému Študent a posielanie ich do CDO
F2. asynchrónne prijímanie odpovede a importovanie do systému Študent
F3. prezeranie stavu komunikácie
F4. zobrazenie nápovedí[31]

Na každej fakulte je určená osoba (vo väčšine prípadov kreditová referentka študijného oddelenia), na počítači ktorej bol okrem štandardnej inštancie systému Študent nainštalovaný aj adaptér XCDO.

Spustenie adaptéra XCDO je zabezpečené vždy štarte operačného systému Win98/Win2000/WinXP[32]. Automaticky raz, každý pracovný deň po prvom zapnutí adaptér vykoná funkciu F1 (export údajov a poslanie do CDO). Súčasne je spustená aj funkcia F2 (čakanie na odpoveď), ktorá beží neustále a ktorá po prijatí správy zaistí jej naimportovanie do systému Študent. Používateľ je informovaný pred začatím posielania/prijímania správy, ako aj pred exportovaním/importovaním údajov. Používateľ počítača môže kedykoľvek vypnúť adaptér XCDO, spustiť ho znova, a požiadať o vykonanie funkcií F1, F3, F4. Vypnutie F2 znamená súčasne ukončenie činnosti adaptéra XCDO, na čo je používateľ vhodnou formou upozornený. Pred skončením adaptér XCDO kontroluje, či práve neprebieha posielanie a keď áno,

[31] V nápovedi používateľ nájde stručný popis systému a kontakt na správcu.
[32] Môžeme predpokladať, že počas pracovnej doby tento adaptér beží neustále, aj keď nemôžeme vylúčiť, že používateľ ho z nejakého dôvodu nevypne.

tak počká na dokončenie, v inom prípade skončí okamžite[33]. Indikácia stavu a prístup používateľa k funkciám adaptéra sa realizuje prostredníctvom „system tray“.

7.2.1 Zobrazenie stavu systému

Na zobrazenie stavu systému sa používa samostatné okno (pop-up) alebo tzv. Balloon tip. Rozlišujeme nasledovné typy hlásení:

SI. informácia (napr.: úspešný export, práve prebieha posielanie, ...)
SII. upozornenie (prebieha export[34], import)
SIII. varovanie (niektorá činnosť, napr. posielanie, export, a podobne zlyhala, avšak adaptéru sa podarilo o tejto udalosti informovať správcu systému)
SIV. ERROR (vážna chyba v systéme, treba informovať správcu osobne)

7.2.2 Prezeranie stavu systému (F3)

Prezeranie stavu systému slúži pre používateľa ako prvotný zdroj informácií o úlohách doteraz vykonaných adaptérom XCDO. Obsahuje dve časti:

- zoznam zobrazených okien s dátumom, časom a textom (boli zvýraznené inou farbou podľa typov hlásenia),
- súhrnné informácie – tabuľka so stĺpcami dátum a čas, meno používateľa, a znázornenie, či bol úspešný príslušný cyklus spracovania údajov (t.j. poslanie údajov do CDO a príjem zodpovedajúcej odpovede – toto vlastne predstavuje atomickú operáciu z pohľadu používateľa). Rozlišujú sa nasledujúce stavy:
  - chyba: nepodarilo sa spraviť export,
  - prebieha posielanie (N. pokus),
  - posielanie neúspešné,
  - posielanie úspešné, odpoveď zatiaľ neprijatá,
  - odoslanie a príjem ukončený.

  Pre každý stav sa uvádza čas, kedy sa tento stav nastal.

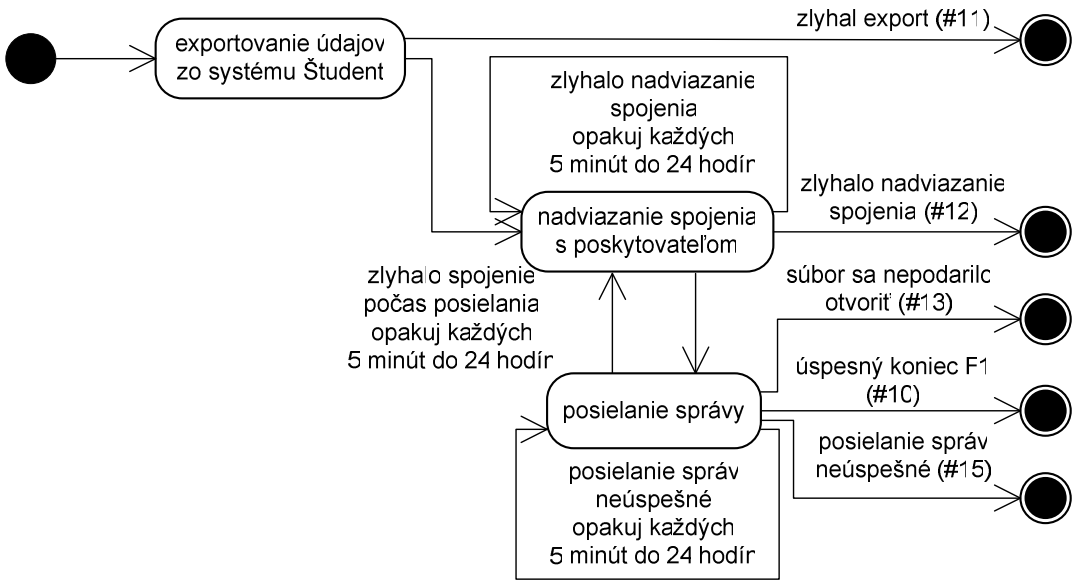
Pozrime si na funkcionalitu adaptéra XCDO podrobnejšie.

7.3 Spolupráca poskytovateľa JMS, adaptéra XCDO a systému Študent

7.3.1 Exportovanie údajov zo systému Študent a posielanie správy do CDO (F1)

Stavový diagram pre F1 vyzerá nasledovne:

[33] Ak práve prebieha prijímanie, je ukončené okamžite s tým, že správy ostanú v rade JMS.
[34] Môže značne spomaliť výkon pracovnej stanice.



Obrázok 7-4: Stavový diagram pre funkcionalitu F1

Po spustení F1 najprv prebieha exportovanie údajov do súboru, pri neúspechu vrátime riadiacemu vláknu chybu s číslom 11, „zlyhal export“. Po úspešnom exporte nasleduje nadviazanie spojenia s poskytovateľom JMS, pri neúspechu vrátime chybu „zlyhalo nadviazanie spojenia“ (#12) a skúsime sa pripojiť po piatich minútach opäť (toto skúšame počas 24 hodín, po ktorých vyhlásime pripojenie za neúspešné). Po úspešnom nadviazaní nasleduje posielanie správy. Tu sa môžu nastať nasledovné situácie:

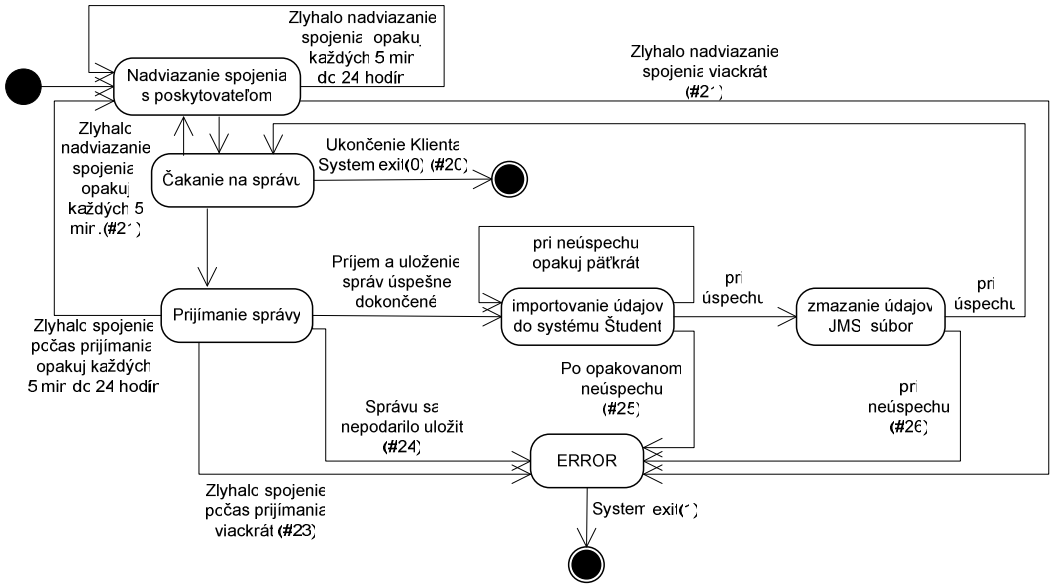
- súbor s exportovanými údajmi sa nepodarilo otvoriť (#13),
- zlyhalo spojenie počas posielania (#14) – opakovať obdobne ako pri #12,
- posielanie správ neúspešné (#15) – opakovať obdobne ako pri #12,
- úspešný koniec (#10).

Pri zlyhaní spojenia sa pokúsime znova nadviazať spojenie a po neúspešných pokusoch do 24 hodín vrátime chybu #15. Keď posielanie bolo úspešne dokončené, vrátime #10.

Pri akomkoľvek neúspechu treba notifikovať správcu.

7.3.2 Asynchrónne prijímanie odpovede a importovanie do systému Študent (F2)

Stavový diagram pre F2 vyzerá nasledovne:



Obrázok 7-5: Stavový diagram pre funkcionalitu F2

Po spustení F2 sa nadviaže spojenie s poskytovateľom JMS asynchrónnym spôsobom a prejdeme do stavu čakania na správu (môže ísť o odpoveď predtým poslaných údajov alebo o správu generovanú CDO ako odpoveď na poslanie údajov z niektorej inej fakulty). Keď je v prislúchajúcom rade k dispozícii správa, začína sa príjem a uloženie správy (používa sa pritom transakčné spracovanie). Pokiaľ sa na ľubovoľnom mieste nastane chyba, používateľ aj správca je o tom informovaný a postup je opakovaný po každej piatej minúte do 24 hodín. Keď ani do 24 hodín sa nepodarí proces korektne ukončiť, dostaneme sa do stavu ERROR. Tento stav znamená, že sme sa dostali do takej situácie, ktorá sa dá riešiť len so zásahom správcu (ktorého XCDO informuje o poruche), a následne je adaptér XCDO ukončený, aby uvoľnil zdroje počítača.

Po korektnom prijímaní a uložení správ začína importovanie údajov do systému Študent, pri neúspechu je správca informovaný a importovanie údajov je opakované ešte trikrát. Po opakovanom neúspechu sa dostaneme do stavu ERROR. Pri úspechu zmažeme správu z radu (zavoláme funkciu commit) a zmažeme aj dočasný súbor. Používateľa informujeme o korektnom ukončení, príjme a importovaní údajov a dostaneme sa do stavu čakania. Pri neúspechu informujeme používateľa aj správcu a prejdeme do stavu ERROR (zavoláme funkciu rollback).

7.3.3 Funkcia F1 – Technické detaily

Exportovanie údajov prebieha zavolaním externej aplikácie xcdo.exe, viď. prílohu B – Popis rozhrania medzi XCDO a xcdo.exe

Zobrazenie chybových hlásení zabezpečí riadiace vlákno adaptéra XCDO, modul realizujúci F1 ho len informuje o stave systému cez logovací mechanizmus.

Ďalšie vlastnosti:

- modul realizujúci F1 je spustený ako samostatné vlákno,
- všetko loguje.

7.3.4 Funkcia F2 – Technické detaily

Importovanie údajov prebieha zavolaním externej aplikácie xcdo.exe, viď. prílohu B – Popis rozhrania medzi XCDO a xcdo.exe

Zobrazenie chybových hlásení podobne ako v prípade funkcie F2 zabezpečuje riadiace vlákno adaptéra XCDO, modul realizácie F2 len ho informuje o stave systému cez logovací mechanizmus.

Ďalšie vlastnosti:

- modul realizujúci F2 je spustený ako samostatné vlákno,
- všetko loguje,
- dáta v JMS nechá, pokiaľ nie je presvedčený, že import bol úspešný.

7.4 Informovanie správcu

Informovanie správcu prebieha dvomi spôsobmi:

- do logu uloženého na serveri (počítame s tým, že spojenie na server obsahujúci systém Študent je spoľahlivé),
- k tomu ešte mailom[35] (je to paralelný kanál, nezávislý od JMS a dokonca od sieťového spojenia na univerzitu) – konkrétne protokolom SMTP na lokálny fakultný mailserver.

Logovanie prebieha s tým istým spôsobom pre F1 aj pre F2. Meno súboru je špecifikované v nastavení. F1 alebo F2 okrem svojho stavu bude pridávať k správe atribút dátum a čas vytvorenia správy, podľa tohto vieme presne identifikovať, či správa poslaná dňa x.y.z v čase u:v:w ešte nebola celkom spracovaná. V logovacom súbore záznamy obsahujú nasledovné atribúty:

- dátum a čas pridania záznamu do logu,
- dátum a čas vytvorenia správy ,
- ID správy (poznámka: ID budú robené sekvenčne, budú sa pamätať v súbore[36]),
- meno používateľa kto vyvolal akciu (napr.: “automaticky“ alebo “Janko Hraško“),
- priorita správy: 0 znamená, že je to pre používateľa, 1 – pre správcu, 2 – pre programátora. Napríklad: keď chceme zobraziť záznamy pre správcu, tak vyberieme tie, ktoré majú prioritu 0 a 1,
- typ záznamu: OK – úspešné (SI, SII), CHYBA – neúspešné (SIII), VAZNA_CHYBA (SIV) – vážna situácia, kedy treba zvýrazniť záznam,

[35] Možnosť posielania logu xcdo.exe.
[36] môže teoreticky slúžiť aj na zaistenie, že posielanie robí len 1 program v danej chvíli (uzamykaním) v druhej verzii XCDO

- slovný popis záznamu v nasledujúcom riadku resp. riadkoch,
- s tým, že riadok v slovnom popise nemôže začínať „//“, „/*“, „*/“ ani „[“ a „]“,
- v logu je možné pridať aj komentáre začínajúce // alebo blok začínajúci /* a končiaci */ – takto môžeme aj zvýrazniť niektoré správy.

príklad *.log súboru:

```
[7.3.2005:9:21:13] ID: 367 (7.3.2005:9:00:13) automaticky 0 OK

Správa bola úspešne odoslaná

[8.3.2005:12:10:57] ID: 368 (8.3.2005:12:09:13) Janko Hraško 1 VAZNA_CHYBA

// ========================================================

Správu nebolo možné odoslať, zlyhalo spojenie, nepodarilo sa informovať správcu

// ========================================================
```

7.5 Študijné oddelenie bez pripojenia do univerzitnej siete

V prípade právnickej fakulty treba upraviť F1 aj F2 tak, aby sa XCDO správal nasledovne:

- na počítači pripojenom do univerzitnej siete bude nasadený XCDO (PC s internetom),
- na počítači pripojenom do siete študijného oddelenia bude nasadený xcdo.exe (PC so systémom Študent),
- medzi nimi bude komunikácia realizovaná cez prenosové médium (napríklad na diskete),
- komunikácia offline treba zabezpečiť tak, aby používateľa zaťažovala čo najmenej. T.j. nemôžeme predpokladať, že pri každom prijatí nových údajov používateľ bude s prenosovou médiou prechádzať medzi dvoma počítačmi. Počet prijatých správ pohybuje okolo desať za deň a treba zabezpečiť aj denné automatické posielanie údajov do CDO so spoluprácou používateľa,
- treba používateľa vhodným spôsobom informovať o tom, čo má robiť s prenosovým médiom..

7.5.1 komunikácia offline – Technické detaily

Komunikácia offline bude vyzerať z požívateľského hľadiska nasledovne:

- XCDO (PC s internetom) požiada po prvom zapnutí počítača, resp. raz denne používateľa o spoluprácu na vloženie prenosového média vyhradeného pre tento účel do príslušnej mechaniky,

- následne XCDO uloží potrebné údaje[37] a skriptovací/dávkový (BAT) súbor na prenosové médium. Požiada používateľa o presun prenosového média do druhého počítača a spustenie BAT súboru[38],
- po vložení prenosového média, spustení BAT súboru budú vyrobené na prenosovom médiu logovacie súbory obsahujúce úspešnosť/neúspešnosť jednotlivých krokov. Nakoniec bude používateľ požiadaný o presun prenosového média znova ku počítaču obsahujúceho XCDO,
- po vložení prenosového média do príslušnej mechaniky XCDO kontroluje logovacie súbory a vyhlási komunikáciu offline (resp. jej jednotlivé kroky) za úspešnú alebo neúspešnú.

BAT súbor má za úlohu vykonať export údajov a následný import. Musí dodržiavať nasledovné pravidlá:

- musí vždy vykonať export údajov a uložiť ich na prenosové médium s logovacím súborom o úspešnosti,
    - v prípade neúspechu zapíše neúspech do logovacieho súboru a skončí.
- následne v dobrom poradí[39] vykoná postupné importovanie údajov a loguje prácu,
    - v prípade prvého neúspechu nepokračuje v importovaní zvyšných údajov a zapíše neúspech do logovacieho súboru,
- vždy informuje používateľa o skončení práce a poprosí ho o presun prenosového média.

XCDO treba upraviť tak, aby vedel prijať všetky správy, ktoré sú v prislúchajúcom rade dostupné (správy v rade musí nechať), podľa počtu a typov správ vyrobil BAT súbor na vykonanie potrebných príkazov. Po komunikácii offline musí overiť, ktoré kroky boli vykonané:

- po úspešnom vykonaní exportovania údajov musí získané údaje poslať do CDO,
    - v prípade neúspechu informovať používateľa,
- po úspešnom vykonaní importovania údajov z prislúchajúceho radu správ má vymazať a informovať používateľa o úspešnosti,
    - v prípade neúspechu vymazať z prislúchajúceho radu správ, ktoré sa podarilo importovať, zvyšné nechať na nasledujúce komunikácie a informovať používateľa o neúspechu.

[37] Pravdepodobne bude treba spakovať údaje v prípade použitia diskety
[38] Je možné vytvoriť odkaz na súbor napríklad na pracovnú plochu na uľahčenie práce používateľa
[39] Poradie je podľa poradia radu, t.j. podľa ID správy

7.6 Princíp komunikácie pre CDO-VIKUK a CDO-PS

Táto podkapitola popisuje riešenie problému komunikácie medzi CDO a VIKUK. Obdobným spôsobom by mohla byť zabezpečená aj komunikácia CDO-PS. Adaptér zabezpečujúci túto komunikáciu bol nazvaný SynchAdapterVili (Vili je angl. skratka Virtual Library).

SynchAdapterVili synchrónnym spôsobom nadviaže spojenie s poskytovateľom JMS, pozrie, či je v prislúchajúcom rade nová správa a keď je k dispozícii, zapíše jej obsah do XML súboru. Po uložení správy začína sa spracovanie XML súboru správcom VIKUK. Keď spracovanie bolo úspešne dokončené, adaptér správu vymaže z radu, v inom prípade ho tam nechá. Ako posledný krok je overené, či v rade je ďalšia (aktuálnejšia) správa, a príslušná informácia je zobrazená na konzole adaptéra SynchAdapterVili.

V každej správe sú k dispozícii informácie o dátume a čase vytvorenia správy.

CDO každý deň, vo večerných hodinách zabezpečuje exportovanie potrebných údajov do príslušného radu. Typ správy je v súčasnosti BytesMessage veľkosti do 10MB, ktorý obsahuje v sebe XML súbor s požadovanou štruktúrou.

Poznámka: v druhej verzii komunikácie je plánované nahradenie tejto komunikácie technikou Request-Reply, o ktorej môžete podrobnejšie dočítať v [Ter06].

7.7 Riešenie bežných prevádzkových problémov

Počas osemmesačnej prevádzky sme sa stretli s nasledujúcimi problémami:

- pri zlyhaní operačného systému MS Windows 98, ktoré sa nastáva relatívne často, ostávalo vytvorené spojenie s poskytovateľom JMS, ktoré spôsobovalo, že adaptér sa po obnovení prevádzky nemohol na príslušný rad pripojiť znovu[40]: riešením bola úprava konfiguračného súboru poskytovateľa JMS (parameter Imq.ping.interval), ktorá zaistila, že „mŕtve“ spojenia boli automaticky ukončené,
- havária servera JMS: po preinštalovaní servera pracovníkmi Centra informačných technológií UK bolo potrebné nainštalovať znovu poskytovateľa JMS a nakonfigurovať ho,
- dolaďovanie XCDO komunikácie offline: počas prevádzky bolo zistené, že počet posielaných správ, ktoré sú odpoveďou na požiadavky z iných fakúlt je relatívne veľký, a preto bolo potrebné optimalizovať komunikáciu,
- zmena nastavení celouniverzitného SMTP servera: vzhľadom na zmenu konfigurácie SMTP na úrovni univerzity bolo potrebné zmeniť nastavenia týkajúce sa adresy servera SMTP slúžiaceho na informovanie správcu,

[40] počet naraz pripojených klientov na jeden rad je obmedzený

- zmena počítača používajúceho XCDO: keďže potrebné súbory sú umiestnené v zdieľanom adresári na serveri, ktorý používa aj systém Študent, bolo v tomto prípade treba len nainštalovať Javu a zabezpečiť automatické spustenie adaptéra po zapnutí počítača,
- zmena používateľa XCDO, osoby: bolo treba zmeniť konfiguračný súbor, konkrétne meno, priezvisko, e-mail a informovať správcu pri zmene telefónneho čísla používateľa,
- zmena miestnosti a/alebo zmena počítača: bolo treba informovať správcu o zmene, aby v prípade potreby návštevy vedel miestnosť a umiestnenie počítača,
- reinštalácia operačného systému: obdobne ako pri „zmena počítača používajúci XCDO“.

8 Záver

Cieľom práce je navrhnúť a čiastočne realizovať technické riešenie pre výmenu informácií medzi vybranými subsystémami informačného systému Univerzity Komenského. Úlohou bolo sformulovať popis problému vrátane konkrétnych požiadaviek na jeho riešenie, navrhnúť riešenie, následne vybrať, resp. vytvoriť potrebné komponenty a vytvorené dielo uviesť do testovacej prevádzky v prostredí informačného systému univerzity.

Tento cieľ sa podarilo splniť. Boli pozbierané základné informácie o jednotlivých subsystémoch a o požiadavkách na ich komunikáciu a na základe nich bolo navrhnuté riešenie. Následne boli sformulované požiadavky na výber produktu. Bol vytvorený zoznam relevantných nekomerčných produktov, ktoré boli následne preskúmané a ich vybrané funkcie aj prakticky otestované.

Na základe vykonaného porovnania sme vybrali konkrétneho poskytovateľa JMS, o ktorom predpokladáme, že je vyhovujúci vzhľadom k sformulovaným požiadavkám. Následne sme vytvorili dve aplikácie na zabezpečenie komunikácie medzi vybratými subsystémami UK. Tieto aplikácie boli nasadené do reálnej prevádzky a s vybraným produktom v súčasnosti zabezpečujú komunikáciu medzi systémom Študent, centrálnou databázou osôb a virtuálnou knižnicou. Počas nasadenia sme získali praktické skúsenosti so spoluprácou so správcami jednotlivých systémov (Študent, CDO, VIKUK) aj používateľmi, ktorí vytvorenú aplikáciu denne používajú.

Ako pokračovanie práce by bolo možné skúmať výkonnosť a kapacitné parametre nekomerčných poskytovateľov JMS. Bolo by možné tiež vytvoriť ďalšie adaptéry zabezpečujúce komunikáciu v rámci informačného systému UK. Zároveň by bolo zaujímavé aj zopakovať testy spoľahlivosti pri niektorých produktoch (ActiveMQ a JBoss Messaging), ktoré boli počas písania tejto práce len vo fáze vývoja.

Zoznam bibliografických odkazov

[Bal06] BALTOVÁ, Alexandra: *Integrácia aplikácií prostredníctvom výmeny správ*. Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2006. (v tlači) – Diplomová práca.

[Bis06] BISTÁK, Ľuboš: *Technológie pre webové služby*. Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2006. (v tlači) – Diplomová práca.

[Codeh] *ActiveMQ* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://activemq.codehaus.org>

[Corid] *MantaRay* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.coridan.com/BuildaGate5/general2/data_card.php?U=no&%20SiteName=coridan&ItemID=89222120&ValuePage=Card10>

[Cri03] *High-Performance JMS Messaging, A Benchmark Comparison of Sun Java System Message Queue and IBM WebSphere MQ* [online]. Crimson Consulting Group, Inc., 2003. Formát PDF. Dostupné na internete: <http://wwws.sun.com/software/products/message_queue/wp_JMSperformance.pdf>

[GDi06] *Google Directory: Java Message Service* [online]. Domovská stránka JMS. Formát HTML. Dostupné na internete: <http://www.google.com/Top/Computers/Programming/Languages/Java/Server-Side/Java_Message_Service>

[Gre04] HOHPE, G., WOOLF, B. *Enterprise Integration Patterns, Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2004. ISBN 0-321-20068-3

[Hermes] *Hermes JMS* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.hermesjms.com>

[Chr00] BRITTON, CH. *IT Architectures and Middleware, Strategies for Building Large, Itegrated Systems.* Addison-Wesley, 2000. ISBN 0-201-70907-4

[IEE93] *IEEE Std 830-1993 IEEE Recommended Practice for Software Requirements Specifications - Description* [online].1993. Domovská stránka. Špecifikácia štandardu. Formát HTML. Dostupné na internete: <http://standards.ieee.org/reading/ieee/std_public/description/se/830-1993_desc.html>

[Inf03] BABCOCK, CH. *Tibco Sues Sonic Over Product Comparison* [online]. InformationWeek, Sept. 2003. Formát HTML. Dostupné na internete: <http://www.informationweek.com/story/showArticle.jhtml?articleID=14200324>

[JBASD] *JBoss Application Server Documentation Library* [online]. Domovská stránka dokumentácie. Formát HTML. Dostupné na internete: <http://labs.jboss.com/portal/jbossas/docs>

[JBMes] *JBoss Messaging* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.jboss.com/products/messaging>

[JBoMQ] *JBossMQ* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.jboss.org/wiki/Wiki.jsp?page=JBossMQ>

[Jnd99] *Java™ Naming and Directory Interface* [online]. Verzia 1.2. Sun Microsystems, Júl. 1999. Domovská stránka. Špecifikácia štandardu. Formát PDF. Dostupné na internete: <java.sun.com/products/jndi>

[JORAM] *JORAM: Java™ Open Reliable Asynchronous Messaging* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://joram.objectweb.org>

[JTA02] *Java™ Transaction API (JTA) Specification* [online]. Verzia 1.0.1B. Sun Microsystems, Nov. 2002. Špecifikácia technológie. Formát PDF. Dostupné na internete: <http://java.sun.com/products/jta>

[Kri04] *JMS Performance Comparison, Performance Comparison for Publish Subscribe Messaging* [online]. Krissoft Solutions, Okt. 2004. Formát PDF. Dostupné na internete: <http://www.fiorano.com/comp-analysis/jms_perf_report.htm>

[KS06] *Katalóg služieb Integrovaného informačného a komunikačného systému Univerzity Komenského – návrh*, interný materiál Centra informačných technológií Univerzity Komenského, 2006

[Log05] *ActiveMQ Performance* [online]. Jún. 2005. Domovská stránka výkonnostného testu nekomerčných produktov. Formát PDF. Dostupné na internete: <http://activemq.codehaus.org/Performance>

[MQD05] *Sun Java™ System Message Queue Documentation* [online]. Domovská stránka dokumentácie. Formát HTML. Dostupné na internete: <http://docs.sun.com/app/docs/coll/1307.1>

[Nat01] SANDLAND, NATHAN, FAKER. *An Architectural Analysis of Message Oriented Middleware* [online]. 2001. Formát HTML. Dostupné na internete: <http://wiki.cs.uiuc.edu/cs427/Messaging+Systems>

[Obj05] *ObjectWeb* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.objectweb.org>

[Ope06] *The Open Group* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.opengroup.org>

[OpenJ] *OpenJMS* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://openjms.sourceforge.net>

[QoS04] CHEN, S., GREENFIELD, P. *QoS Evaluation of JMS: An Empirical Approach* [online] Proceedings of the 37th Hawaii International Conference on System Sciences, 2004. Formát PDF. Dostupné na internete: <http://csdl.computer.org/comp/proceedings/hicss/2004/2056/09/205690276b.pdf>

[Ric01] MONSON-HAEFEL, R. *Enterprise JavaBeans, 3rd Edition.* [online]. O'Reilly Online Catalog, Sept. 2001. ISBN: 0-596-00226-2, Formát HTML. Dostupné na internete: <http://www.oreilly.com/catalog/entjbeans3/chapter/ch13.html#_Toc495111930>

[Sha02] SHAUN, T. *Enterprise JMS programming.* M & T Books, 2002. ISBN 0-7645-4897-2

[SJSMQ] *Sun Java™ System Message Queue* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.sun.com/software/products/message_queue/index.xml>

[SMQEE] *Sun Java™ System Message Queue Enterprise Edition* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.sun.com/software/products/message_queue_ee>

[Som02] SOMMERVILLE, I. *Szoftverrendszerek fejlesztése [Software engineering]*. Panem, 2002. ISBN 963 545 311 6

[Son03] *JMS Performance Comparison: Publish/Subscribe Messaging* [online]. Sonic Software Corporation, Nov. 2003. Formát PDF. Dostupné na internete: <http://www.sonicsoftware.com/products/whitepapers/docs/jms_comparison_tibco.pdf>

[Sun02] *Java™ Message Service Specification* [online]. Verzia 1.1. Sun Microsystems, Apr. 2002. Špecifikácia technológie. Formát PDF. Dostupné na internete: <http://java.sun.com/products/jms/docs.html>

[Ter06] TERKANIČ, Ján. *Zdieľanie informácií o osobách v informačnom systéme Univerzity Komenského*. Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2006. (v tlači) – Diplomová práca.

[XOp91] *Distributed Transaction Processing* [online]. XA X/Open Company Ltd. December 1991. Špecifikácia štandardu. Formát PDF. Dostupné na internete: < http://www.opengroup.org/onlinepubs/009680699/toc.pdf >

Prílohy

A – Popis rozhrania medzi XCDO a CDO

Smerom k CDO správa je inštanciou typu `javax.jms.ByteMessage`, ktorá obsahuje navyše:

- „`adapterFS_IDmessage`“[41] (`int`) – identifikačné číslo správy pre aktuálnu fakultu (unikátnosť je zaručená v rámci fakulty, ale na rôznych fakultách môžu byť rovnaké ID),
- „`adapterFS_IDmessage_TimeStamp`“ (`long`) – dátum vytvorenia identifikačného čísla správy,
- „`adapterFS_kod_fakulty`“ (`String`, dĺžka 4),
- „`adapterFS_rocnik`“ (`String`, inicializačnú hodnotu získa adapterFS zo súboru AdapterFS.properties, záznam „`Student.rocnik`“:, začiatočná hodnota: „`2005`“),
- **`getJMSReplyTo`**`()` (`javax.jms.Destination`) – vráti aktualizovaný ukazovateľ na prislúchajúci rad `Queue`, kde treba poslať odpoveď tejto správy.

Smerom k XCDO správa je inštanciou typu `javax.jms.ByteMessage`, ktorá obsahuje navyše:

- „`adapterFS_IDmessage`“ (`int`) – identifikačné číslo správy na ktorú prichádza odpoveď (v prípade, že takáto správa ešte neexistuje, základná hodnota 0),
- „`adapterFS_IDmessage_TimeStamp`“ (`long`) – dátum vytvorenia identifikačného čísla správy na ktorú prichádza odpoveď (ak správa ešte neexistovala, základná hodnota 0),
- „`funkcia`“ (`String`) (štandardne „import“) – znamená s akým parametrom treba zavolať XCDO,
- „`rocnik`“ (`String`, s dĺžkou 4) (základná „2005“) – znamená s akým parametrom treba zavolať XCDO,
- „`posledna_sprava`“ (`boolean`) – v súčasnosti sa tento parameter nepoužíva.

[41] adapterFS je pôvodná skratka pre adaptér fakultnej inštancie systému Študent (XCDO)

B – Popis rozhrania medzi XCDO a xcdo.exe

Komunikácia medzi XCDO a xcdo.exe je zabezpečené cez prenos údajov cez zdieľané súbory a riadené nasledujúcimi exit kódmi:

- 0 – úspešný import/initimport (súbor s údajmi nebol zmazaný), export (bol vytvorený korektný súbor podľa požiadaviek CDO, vid [Ter06]),
- 1 – xcdo.exe má internú chybu, treba informovať správcu systému Študent,
- 2 – xcdo.exe nevedel získať prístup k databáze (zamknúť), treba opakovať po piatich minútach (trikrát).
- 3 – zatiaľ neboli vytvorené databázy pre požadovaný školský rok, treba informovať správcu systému Študent,
- 4 – (rezervovaný),
- 5 – zlé parametre pri zavolaní xcdo.exe.

Parametrizácie xcdo.exe:

- xcdo export ročník meno_súboru[42],
    - napríklad: xcdo export 2005 export.txt,
- xcdo import ročník meno_súboru,
    - napríklad: xcdo import 2005 msg_0001.txt,
- xcdo initimport ročník meno_súboru,
    - napríklad: xcdo initimport 2005 init_imp.txt.

Exit kódy s číslom 1, 3, 5 znamenajú nutnosť zásah správcu systému Študent alebo správcu komunikácie, preto v týchto prípadoch je vyžadujúce skončiť aplikáciu XCDO.

Poznámka: V niektorých prípadoch sa stáva (po skončení vráti exit kód 1), že xcdo.exe čaká na stlačenie klávesnice a zobrazuje červené 'okienko' s príslušnou hláskou. Toto čakanie spôsobí pod operačným systémom využitie procesora na 100 % a následne zahltenie[43].

Predpoklady korektného fungovania xcdo.exe

Pred spusteným xcdo.exe treba overiť upravenie štruktúr databáz v systéme Študent (toto upravenie vykonal správca systému Študent pred začiatkom akademického roku 2005/06).

---

[42] meno_súboru je vo formáte 8.3 a malo by mať príponu .txt pre ľahšie pozretie používateľom XCDO.

[43] Pri spúšťaní externého procesu v jave, konkrétne xcdo.exe nevieme korektne ukončiť pod operačným systémom Win98 (process.destroy()). Z tohto faktu vyplýva, že je potreba zásah používateľa, preto bolo vhodné informovať používateľov XCDO o tom, že sa takáto situácia môže nastať, a aby stlačili ľubovoľnú klávesnicu keď v DOS-ovskom okne uvidia chybové hlásenie. Pod Win2000 a WinXP korektne funguje ukončenie externého procesu a ani DOS-ovské okno nie je zobrazené.

Poznámky:

Nové databázy pred začatím nového školského roku sú vytvorené pri prvom vstupe do menu 'zapis' v systéme Študent. Meno súboru obsahujúcecho databázu pre príslušnú fakultu a ročník je sto11_05.dbf a ďalšie. Po vytvorení potrebných databáz pre požadovaný ročník xcdo.exe vráti prázdny súbor s hlavičkou (nakoľko nebol zapísaný ani jeden študent).

Na testovanie spolupráce xcdo.exe a XCDO je možné využiť prázdny súbor pri funkcionalite import a initimport. Importovanie a export údajov na najväčšej fakulte odhadujeme do 15 minút (pri zápise do databázy Študenta nezamyká celú databázu, len príslušný riadok tabuľky, čo je príčinou relatívne pomalého spracovania).

C – Používateľská príručka pre aplikáciu XCDO

Úvod:

Aplikácia XCDO je program na výmenu informácií medzi systémom Študent a centrálnou databázou osôb (CDO).

Systém Študent je aplikácia, s ktorou bežne pracujete (databáza študentov).

Centrálna databáza osôb (CDO) obsahuje zoznam osôb a ich preukazov. Je umiestnená na rektoráte a slúži ako zdroj údajov pre univerzitné validačné terminály, pre posielanie údajov spoločnosti EMcard (správca informačného systému pre DPB, SAD a ŽSR – samozrejme len pre tých študentov, ktorí podpísali súhlas s poskytnutím osobných údajov) a pre ďalšie univerzitné aplikácie.

XCDO má na starosti automatickú komunikáciu systémov Študent a CDO, teda konkrétne:

1. exportovanie údajov zo systému Študent a ich poslanie do CDO,
2. prijímanie údajov z CDO a ich importovanie do systému Študent.

Exportovanie údajov je proces získania údajov zo systému Študent (ide o údaje o študentoch a ich preukazoch).

Importovanie údajov je proces nahrávania údajov z CDO do systému Študent (ide o údaje o výsledku aktivácie preukazov).

Aktivácia preukazu je priradenie preukazu konkrétnemu študentovi. Aktivácia môže byť úspešná alebo neúspešná. O výsledku aktivácie sa dozviete po prijatí správy z CDO, a to v systéme Študent priamo v karte študenta (SHIFT+F4) – pozrite časť najdôležitejšie. Až po úspešnej aktivácii preukazu môže študent vykonať jeho validáciu na univerzitnom validačnom termináli.

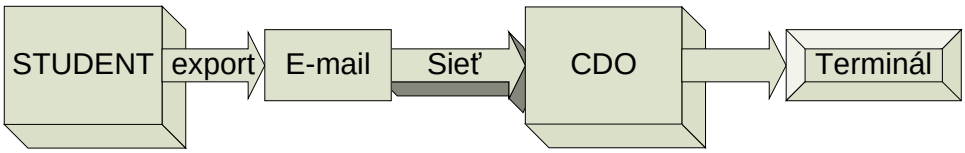
Na komunikáciu medzi systémom Študent a CDO sa používa **univerzitná počítačová sieť** – za bežných okolností ide o špeciálny komunikačný protokol, v prípade výskytu chyby sa použije elektronická pošta na posielanie chybových hlásení administrátorovi systému.

Pôvodný systém (prevádzkovaný v akademickom roku 2004/2005)

V pôvodnom systéme prebiehala komunikácia takto:

1. údaje ste exportovali zo systému Študent ručne
2. výstupný súbor ste poslali elektronickou poštou správcovi
3. správca údaje vložil do CDO, stadiaľ nahral do validačných terminálov a vám poslal odpoveď
4. až po prijatí odpovede ste si mohli byť istí, že údaje sú aktualizované vo validačných termináloch

Graficky môžeme komunikáciu znázorniť nasledovne:



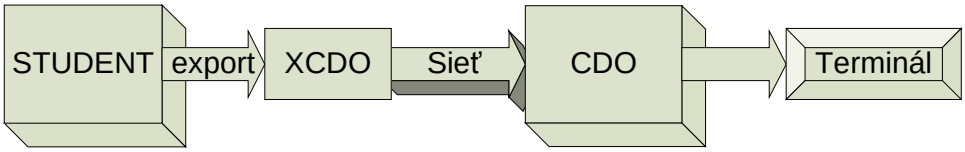
Vidíme, že prvé dva kroky (export a posielanie elektronickou poštou) bolo potrebné vykonať ručne.

Nový systém

Komunikácia prebehne podobným spôsobom, ale:

+ nemusíte ručne exportovať údaje zo systému Študent a posielať ich správcovi (toto vykoná aplikácia XCDO)
+ odpoveď dostanete hneď, ako je to možné
+ odpoveď bude obsahovať aj podrobnú stavovú informáciu o úspechu/neúspechu aktivácie jednotlivých preukazov

Graficky možno komunikáciu znázorniť takto:



Oproti pôvodnému systému sa zmenil spôsob prenosu údajov (z ručného posielania súboru na automatické posielanie aplikáciou XCDO). Automatické posielanie prebieha v pravidelných intervaloch, bez nutnosti zásahu používateľa.

Typy hlásení

Keďže ste v pôvodnom systéme vykonávali jednotlivé operácie ručne, vedeli ste, v ktorom kroku sa nachádzate. Teraz je prenos údajov automatický, preto bolo potrebné vytvoriť množinu stavových správ, ktorých cieľom je informovať vás ako používateľa systému o stave, v ktorom sa systém nachádza.

Aby bolo zobrazenie hlásení čo najprehľadnejšie, zaviedli sme 4 typy hlásení, usporiadaných od informačných správ ku vážnejším hláseniam:

1. informácia (napr. úspešný export, práve prebieha posielanie, ...)
2. upozornenie (prebieha export, import)
3. chyba (posielanie resp. export zlyhal, ale administrátor bol informovaný)
4. vážna chyba (vážna chyba v systéme, treba informovať administrátora osobne)

Prvé hlásenie (informácia) hovorí o tom, v ktorom stave je aplikácia XCDO. V tomto prípade nemusíte nič robiť, hlásenie vás len informuje o tom, čo sa deje.

Prax používania aplikácie XCDO vyžiadala, aby sme vytvorili hlásenie podobné informácii, ktoré sme nazvali upozornenie. Ani v tomto prípade nemusíte nič robiť, hlásenie hovorí o tom, že váš počítač pravdepodobne začne náročnú operáciu, preto bude pracovať pomalšie, než je obvyklé. Takýto typ správy uvidíte pri exporte a pri

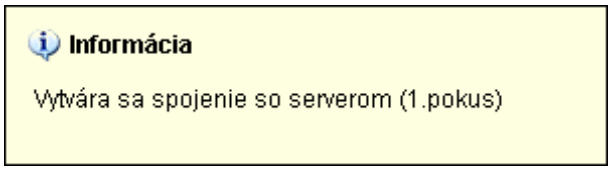
importe údajov. Keď uvidíte toto hlásenie (žltá farba), odporúčame počkať, kým prebehne export resp. import údajov.

Prvé dve hlásenia sú zobrazené 10 sekúnd, potom zmiznú automaticky.

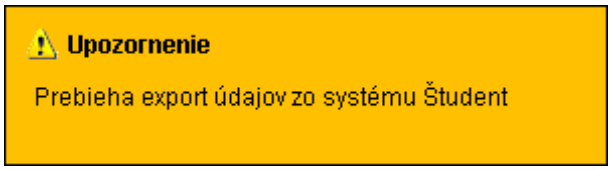
Môže sa nastať situácia, že sa vyskytne chyba počas behu aplikácie XCDO. Tieto chyby sa zobrazujú ako hlásenia, pri ktorých treba kliknúť na tlačidlo OK, čím potvrdzujete, že ste si správu prečítali. V systéme je zabudovaný mechanizmus zaisťujúci automatické posielanie chybových hlásení administrátorovi. Podľa toho, či bolo možné informovať administrátora, rozlišujeme hlásenie typu *chyba* (vtedy nemusíte nič robiť, administrátor bol informovaný) alebo *vážna chyba* (vtedy **treba informovať administrátora** podľa pokynov, ktoré zobrazí aplikácia XCDO).

Vzhľad jednotlivých hlásení je takýto:

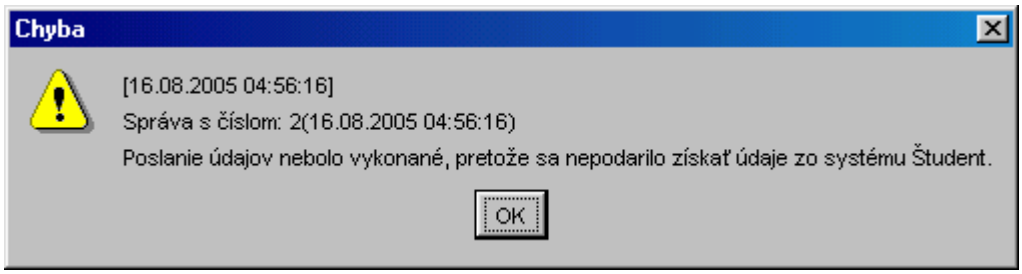
Informácia:



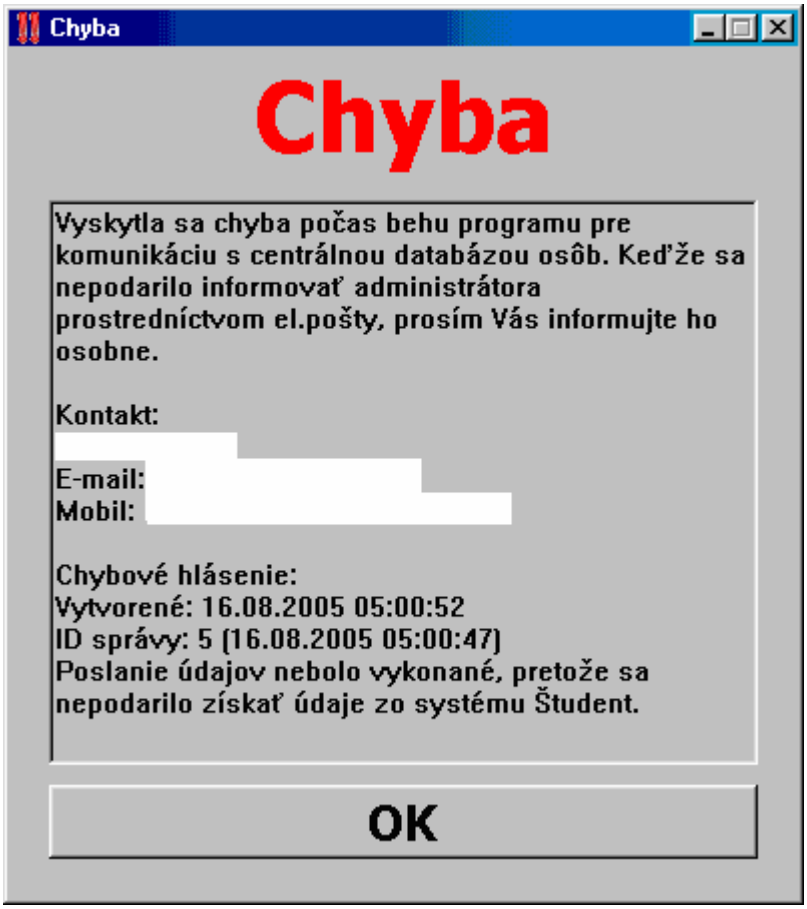
Upozornenie:



Chyba:

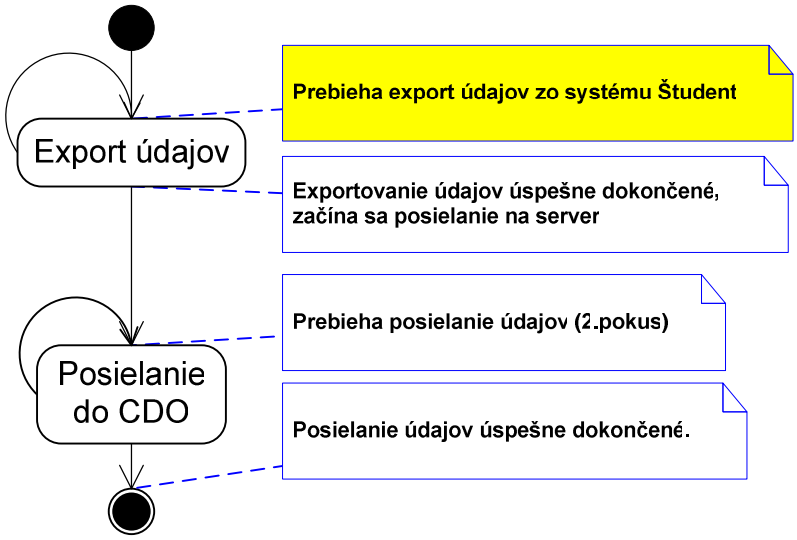


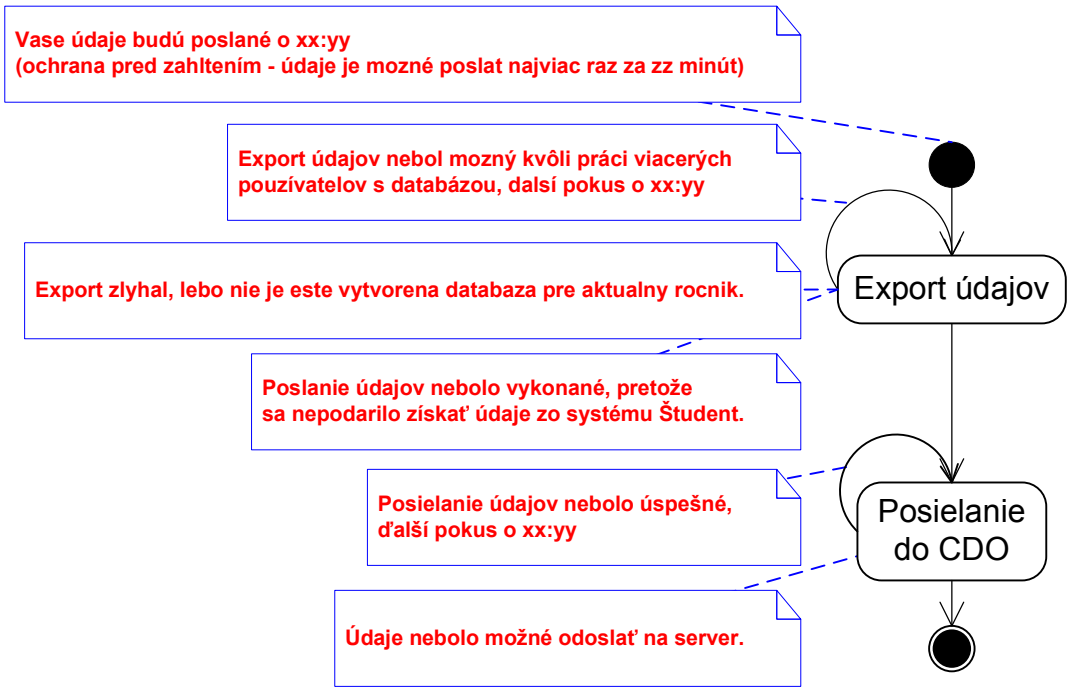
Vážna chyba:



Hlásenia pri posielaní

Aplikácia XCDO pri posielaní údajov musí vykonať dva kroky. Najprv exportuje údaje zo systému Študent a potom získané údaje pošle do CDO. Hlásenia môžete vidieť pred prvým krokom, po vykonaní prvého kroku, pred druhým a po skončení druhého kroku. V prvej časti nasledujúceho obrázku vidíte informatívne hlásenia, v druhej sa nachádzajú chybové hlásenia.



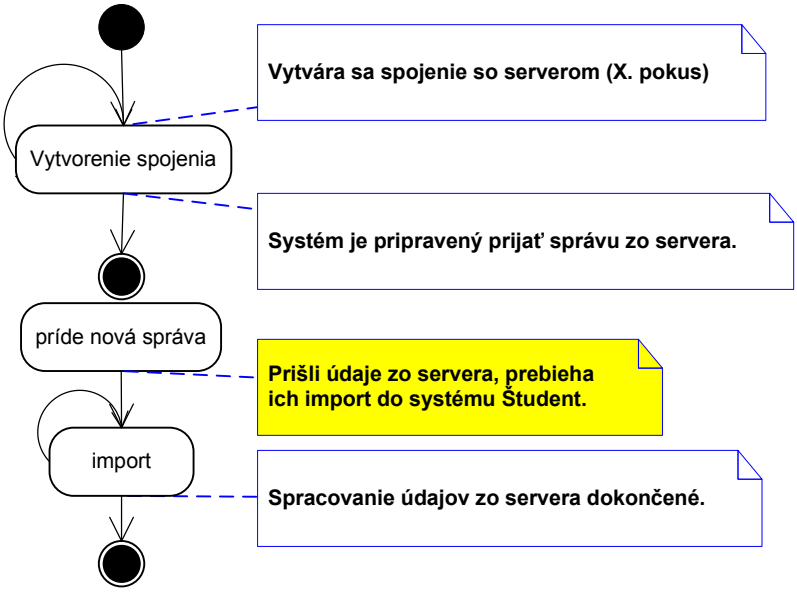


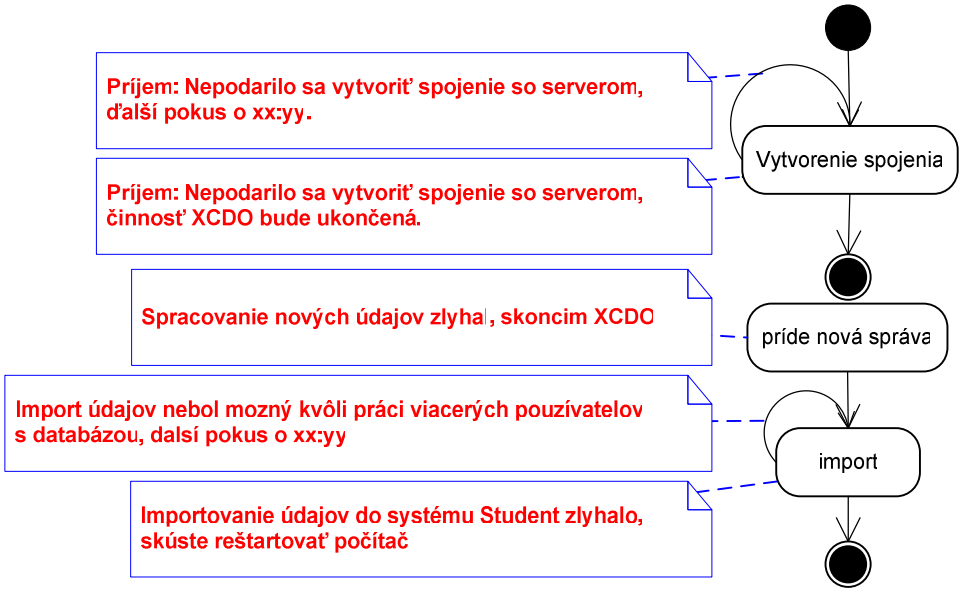
Hlásenia pri prijímaní

Prijímanie údajov z CDO pozostáva z nasledujúcich krokov:

- Prvým krokom je vytvorenie spojenia so serverom, na ktorom beží CDO.
- Po pripojení aplikácia XCDO čaká na príchod údajov zo servera.
- Keď príde nová správa, prebehne automatické importovanie údajov do systému Študent.

Hlásenia, ktoré môžete vidieť, sú:





V prípade, že dostanete správu „Nepodarilo sa vytvoriť spojenie so serverom, činnosť adaptéra bude ukončená“, resp. „Importovanie údajov do systému Študent zlyhalo“, skúste reštartovať váš počítač. Ak sa príjem správ nepodarí zrealizovať ani po reštarte, spojte sa s administrátorom XCDO.

Základné operácie aplikácie XCDO

Keď kliknete pravým tlačidlom myši na ikonu aplikácie XCDO (žlté X s nadpisom CDO) v pravom dolnom rohu obrazovky, môžete vykonať nasledovné akcie:

Pošli

Táto voľba slúži na spustenie posielania údajov v ľubovoľnom okamihu práce. Po stlačení sa text *´Pošli´* zmení na *´zastav posielanie´*, začne sa export údajov a po korektnom exporte systém pošle údaje do CDO.

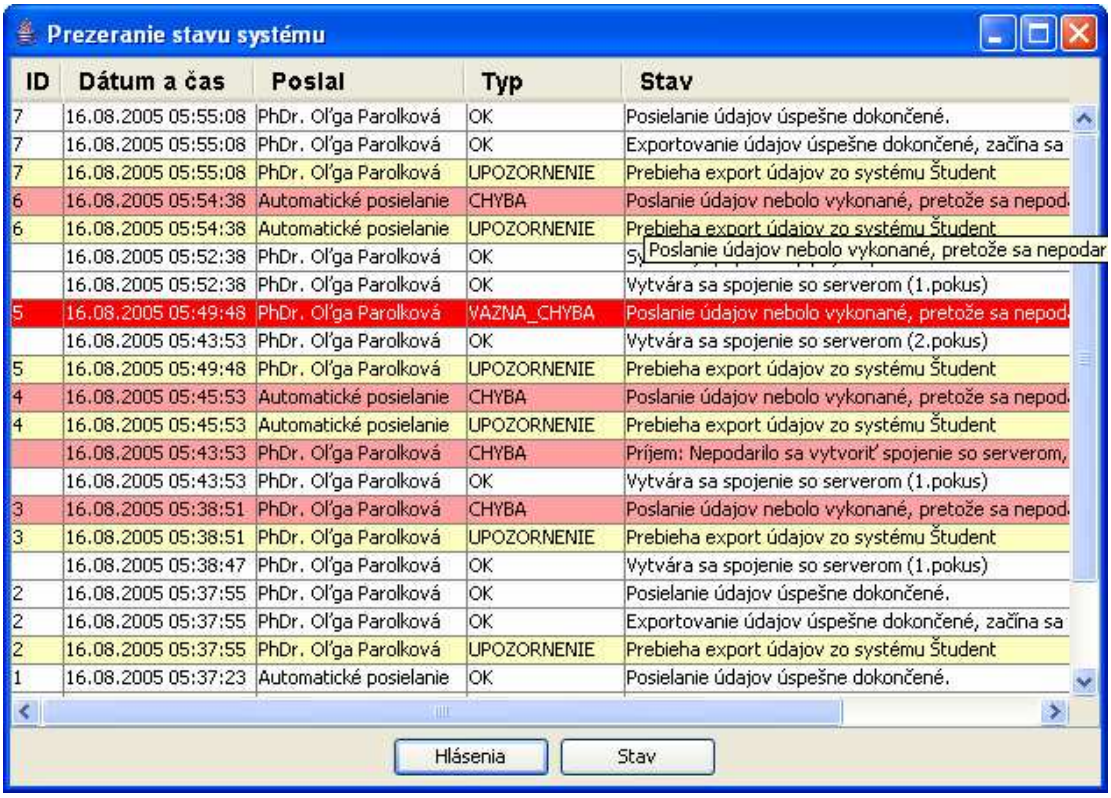
Ak by ste chceli posielanie údajov zastaviť, stlačte *´zastav posielanie´*. Vtedy sa text *´zastav posielanie´* zmení na šedé *´zastavenie´* a musíte počkať, kým prebehne zastavenie posielania. (Zastavenie posielanie však vo väčšine prípadov netreba vykonať – poslaním údajov nespravíte nič zlé, v najhoršom prípade pošlete do CDO tie isté údaje, ktoré ste už raz poslali). Ak by sa vám zdalo, že zastavenie posielania trvá príliš dlho (viac než 15 minút), reštartujte počítač.

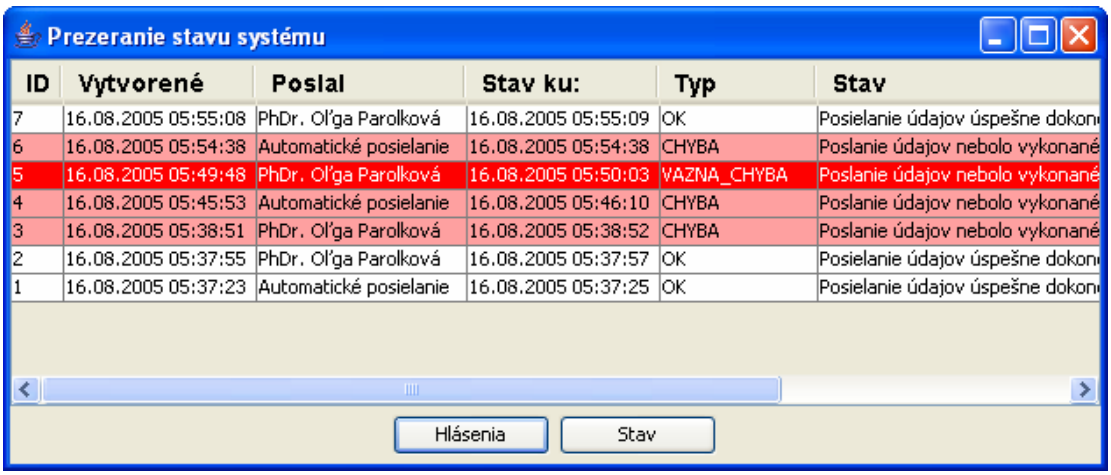
POZOR! Z dôvodu ochrany CDO pred zahltením je možné posielať údaje maximálne raz za 60 minút. Ak spustíte posielanie skôr, systém vás o tom bude informovať, a posielanie automaticky vykoná v najbližšom možnom čase.

Stav systému

(Táto časť návodu bude ešte dopracovaná)

Keď kliknete ľavým tlačidlom myši na ikonku XCDO, zobrazení sa okienko obsahujúce stav systému.





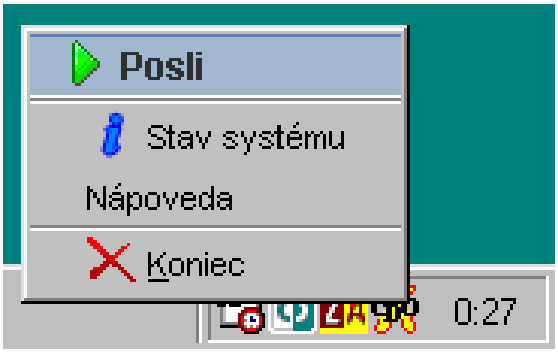
Nápoveda

Slúži na zobrazenie tejto nápovede (tento dokument).

Koniec

Touto funkciou môžete vypnúť aplikáciu XCDO. **Neodporúčame ju používať**, zastavíte tým automatickú komunikáciu s CDO. XCDO sa automaticky spustí pri ďalšom štarte počítača.

Graficky vyzerá menu aplikácie XCDO nasledovne:



Najdôležitejšie

Na fungovanie komunikácie medzi systémom Študent a CDO je potrebné dodržať nasledujúce pravidlá:

- Každý deň, pri prvom zapnutí počítača počkať, kým prebehne exportovanie údajov zo systému Študent (po zobrazení hlásenia *´Exportovanie údajov úspešne dokončené, začína sa posielanie na server´* už môžete pracovať).
- Počas zápisov odporúčame okrem ranného automatického posielania spustiť posielanie aj manuálne, a to po spracovaní aktuálne zapísaných študentov v systéme Študent. Keď na fakulte prebieha zápis napríklad v dvoch kolách (doobedňajšie a poobedňajšie), tak treba spustiť posielanie po skončení doobedňajších aj poobedňajších zápisov (pravý klik myšou na ikonku XCDO → Pošli).
- Posielanie môžete spustiť v ľubovoľnom okamihu. Napríklad, keď sa študent opýta, kedy najskôr môže aktivovať kartu ISIC pri termináli, tak po stlačení ´Pošli´ a po úspešnom poslaní údajov (t.j. keď nebolo zobrazené chybné hlásenie) mu môžete povedať, že po 2 hodinách už môže aktivovať kartu ISIC. (Za predpokladu, že dodržíte interval 60 minút medzi posielaniami, ako je uvedené nižšie.) Samozrejme toto platí za predpokladu, že informácie o príslušnom študentovi ste už vložili do systému Študent.
- Posielať údaje je z dôvodov ochrany CDO pred zaťažením možné najskôr po uplynutí 60 minút od posledného posielania. Keď sa Vám stane, že potrebujete posielať údaje skôr, než o 60 minút, zvoľte funkciu ´Pošli´ a aplikácia XCDO sa postará o to, aby údaje boli poslané v najskoršom možnom termíne. (T.j. netreba opakovať stlačenie ´Pošli´.)
- Keď aplikácia zobrazí veľké okienko so správou, že máte informovať administrátora o chybe, prosím nezabudnite mu zavolať (alebo poslať elektronickú poštu). – Keď sa nedozvie o chybe, nemôže ju opraviť.
- Keď aplikácia zobrazí malé okienko s chybou, znamená to, že nastala chyba, ale administrátor bol o nej informovaný a snaží sa situáciu čím skôr vyriešiť.
- Ostatné hlásenia sú len informatívne. Žlté hlásenie je upozorňovacie a znamená, že počítač bude pravdepodobne pomalší, preto odporúčame prerušiť prácu, kým neuvidíte ďalšie (biele) hlásenie
- Nevypínajte aplikáciu XCDO, lebo potom nebudú odchádzať informácie z vášho systému Študent do CDO a naopak (keby sa vám náhodou podarilo vypnúť aplikáciu XCDO, môžete ju zapnúť tak, že reštartujete počítač)
- Môže nastať situácia, kedy počas exportovania údajov vidíte v okne s čiernym pozadím („*DOSovské okno*“) červené okienko s chybovým hlásením, ktorému nerozumiete. Prosím zapíšte hlásenie na papier a stlačte ENTER. Potom

pravdepodobne dostanete hlásenie od aplikácie XCDO, že sa nepodarilo získať údaje zo systému Študent. Chybové hlásenie, ktoré ste zapísali na papier, pošlite elektronickou poštou na adresu správcu systému Študent (viď kontakty). On chybu opraví hneď, keď to bude možné.

Keď sa príde študent sťažovať, že nemohol na termináli validovať kartu ISIC, tak jednoducho vyhľadáte jeho záznam v systéme Študent a stlačíte SHIFT+F4. Bude vám zobrazené okno, kde nájdete informácie o študentovi a v spodnej časti okna bude záznam, ktorý hovorí o stave aktivácie karty ISIC.

Často kladené otázky

Čo mám robiť, keď počas exportovania údajov vidím v okne s čiernym pozadím (DOSovské okno) červené okienko s chybným hlásením, ktorej nerozumiem?

Prosím napíšte hlásenie na papier a stlačte ENTER. Potom pravdepodobne dostanete hlásenie od aplikácie XCDO, že sa nepodarilo získať údaje zo systému Študent. Chybové hlásenie, ktoré ste zapísali na papier, pošlite elektronickou poštou na adresu správcu systému Študent (viď kontakty).

Čo je to XCDO?

XCDO je skratka pre aplikáciu zabezpečujúci prenos údajov medzi systémom Študent a centrálnou databázou osôb (CDO).

Kde je aplikácia XCDO?

Aplikáciu XCDO nájdete v pravom dolnom rohu obrazovky, ikonka: XCDO (žlté X s nápisom CDO).

Prečo spomaľuje export a import môj počítač?

Lebo XCDO spustí externý DOSovský program na získanie údajov zo systému Študent, čo je náročná operácia.

Prečo je môj počítač taký pomalý, odkedy tu mám nainštalovanú aplikáciu XCDO?

Pravdepodobne máte starší počítač, ktorý má málo pamäti. Prosíme o vašu trpezlivosť. Keď si myslíte, že pri behu aplikácie XCDO je práca na počítači ťažkopádna alebo skoro nemožná, môžete aplikáciu XCDO na vlastnú zodpovednosť vypnúť, ak dodržíte nasledovné pravidlá:

1. Po prvom rannom zapnutí počítača počkáte, kým prebehne automatické posielanie údajov a počkáte cca 20 minút, či príde odpoveď.
2. Potom môžete XCDO vypnúť, ale nezabudnite reštartovať počítač, keď idete na obed (čím necháte naštartovať a bežať aplikáciu XCDO). Nechajte ju bežať, kým sa nevrátite z obeda, resp. aspoň dovtedy, kým neodošle údaje a neprijme odpoveď (do 20 minút).

(Tento problém sa týka len troch fakúlt.)

Kedy koho mám informovať, keď nastane chyba?

Nezabudnite na to, že v celom systéme sú tri časti, kde sa môžu vyskytnúť chyby:

- chyba na strane systému Študent (pravdepodobne vtedy, keď aplikácia XCDO nevie exportovať alebo importovať údaje),
- chyba v komunikácii (keď vidíte veľké okno s chybou, treba informovať administrátora podľa pokynov na obrazovke),
- chyba na strane CDO (keď posielanie údajov prebehlo korektne, ale neprišla odpoveď napríklad do dvoch dní).

V menu aplikácie XCDO po zastavení posielania zostane ´zastavenie´ na dlhší čas, ale potreboval by som poslať údaje, čo mám spraviť?

Treba reštartovať počítač a po reštarte už môžete znova poslať najaktuálnejšie údaje.

Ako môžem zistiť, či už bola aktivovaná karta ISIC pre študenta ´Ferko Mrkvička´?

V systéme Študent vyhľadáte študenta *´Ferko Mrkvička´*. Po stlačení SHIFT+F4 môžete v spodnej časti okna prečítať niektorú z nasledujúcich správ:

- *Požiadavka bola zadaná (3.9.2005 11:13)* – dátum a čas, kedy ste požiadali o aktiváciu preukazu
- *Údaje odoslané → UT (3.9.2005 13:45)* – dátum a čas, kedy boli poslané údaje do CDO

V prípade úspechu:

- *Aktivácia prebehla úspešne (3.9.2005 13:55)*

resp. v prípade, že kartu študentovi aktivuje iná fakulta:

- *Preukaz aktivovaný na základe požiadavky FMFI z dňa 16.5.2005.*

Keď aktiváciu centrálna databáza osôb nevedela vykonať:

- *Chyba: Preukaz č. 1122334455/S14293939284 nie je v databáze vytlačených preukazov.*
- *Chyba: Preukaz č. 1122334455/S14293939284 nie je preukazom denného študenta.*
- *Chyba: Preukaz č. 1122334455/S14293939284 pravdepodobne nepatrí tejto osobe.*
- *Chyba: Preukaz č. 1122334455 (preukaz ext. študenta) nie je možné aktivovať, pretože tento študent nie je študentom externej formy štúdia.*
- *Chyba: Preukaz č. 1122334455/S14293939284 (preukaz ISIC) nie je možné aktivovať, pretože tento študent nie je študentom dennej formy štúdia.*
- *Chyba: Preukaz č. 1122334455 (preukaz ext. študenta) nie je možné aktivovať, pretože tento študent je zároveň študentom dennej formy štúdia na FMFI.*
- *Chyba: Preukaz č. 1122334455/S14293939284 nie je preukazom študenta.*

resp. iné možné chyby.

Poznámka: ako číslo preukazu ostane v prípade chyby pôvodná hodnota (číslo práve aktívnej karty)