ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

Študijný odbor: **Informačné systémy - Aplikovaná informatika**

Bc. Vladimír Smataník

Integrácia služieb prostredníctvom ESB podľa SOA princípov na IBM WebSphere Message Broker platforme

Vedúca: **Ing. Katarína ZÁBOVSKÁ, PhD.**

Reg.č. 334/2012 29. apríla 2013

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program : Informačné systémy

Zameranie: Aplikovaná informatika

Meno a priezvisko	**Osobné číslo**
Vladimír Smataník	553470

Názov práce v slovenskom aj anglickom jazyku

Integrácia služieb prostredníctvom ESB podľa SOA princípov na IBM WebSphere Message Broker platforme

Integration of services by ESB according to principles of the SOA on IBM Web Sphere Message Broker platform.

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:
Diplomová práca bude zameraná na získanie teoretických znalostí a ich priame uplatnenie v praxi pre návrh, vývoj a nasadenie systémov automatizácie obchodných procesov a pravidiel.

Obsah:

1. Oboznámiť sa s princípmi SOA integrácie, integračnými štandardami a referenčnými architektúrami
2. Prakticky zvládnuť nástroj IBM WebSphere Message Broker
3. Navrhnúť komponentný model riešenia
4. Implementovať integračnú časť navrhnutého riešenia v prostredí IBM WebSphere Message Broker
5. Spoločne s tímom prezentovať navrhnuté riešenie integrácie služieb pre projekt
6. Realizovať testovanie a nasadenie riešenia na produkčnú platformu

Témy z predmetov študijného zamerania	5SI030: 1, 2, 5
Meno a pracovisko vedúceho DP:	Ing. Katarína Zábovská, PhD., KMME, ŽU
Meno a pracovisko tútora DP:	

vedúci DP	tútor	vedúci katedry	garant
(dátum a podpis)	(dátum a podpis)	(dátum a podpis)	(dátum a podpis)

Zadanie zaregistrované dňa 29. 10. 2012 pod číslom 334/2012 podpis _____

Abstrakt

BC. SMATANÍK VLADIMÍR: *Integrácia služieb prostredníctvom ESB podľa SOA princípov na IBM WebSphere Message Broker platforme* [Diplomová práca]
Žilinská Univerzita v Žiline, Fakulta riadenia a informatiky, Katedra makro a mikroekonomiky.
Vedúci: Ing. Katarína ZÁBOVSKÁ, PhD.
Stupeň odbornej kvalifikácie: Inžinier v odbore Informačné systémy, Žilina.
FRI ŽU v Žiline, 2013 — 76 s.

Obsahom práce je analýza a návrh systému včasného varovania banky pri defaulte korporátnych klientov.
Práca je zameraná na integráciu jednotlivých častí riešenia, definovanie rozhraní, služieb a prístupov k databáze a dátovému skladu, ako aj samotný návrh konceptuálneho a dátového modelu.
Teoretickým základom sú princípy servisne orientovanej architektúry, integračné štandardy a referenčné architektúry.
Prakticky sa práca zaoberá prostredím platformy IBM WebSphere Message Broker a systémom včasného varovania v častiach návrh a implementácia, od teoretických základov a architektúry až po konečné testovanie funkcionality. Časťou práce je analýza banky a risk manažmentu, návrh dátového modelu a implementácia servisov v nástroji IBM WebSphere Message Broker, navrhnuté číselníky a testovacie dáta.

Kľúčové slová: banka, úver, systém včasného varovania, riziko

Abstract

BC. SMATANÍK VLADIMÍR: *Integration of services by ESB according to principles of the SOA on IBM Web Sphere Message Broker platform* [Diploma thesis]
University of Žilina, Faculty of Management Science and Informatics, Department of macro and microeconomics.
Tutor: Ing. Katarína ZÁBOVSKÁ, PhD.
Qualification level: Engineer in field Information Systems, Žilina.
FRI ŽU v Žiline, 2009 — 76 p.

The content of this diploma thesis is analysis and design of early warning system for default of corporate clients in bank.
Project is aimed on integration of the parts of the solution, defining the interfaces, services and accesses to database and data warehouse and on the design of the conceptual and data model.
Theoretical basis are principles of service oriented architecture, integration standards and reference architectures.
Thesis practically deals with the environment of IBM WebSphere Message Broker and early warning system in design and implementation parts, from theoretical basis and architecture to the final testing of the functionality. Part of the document is analysis of the bank and risk management, design of the data model and implementation of the services in IBM WebSphere Message Broker, created lookup tables and test data.

Keywords: bank, loan, early warning system, risk

Čestné prehlásenie

Prehlasujem, že som túto prácu napísal samostatne, a že som uviedol všetky použité pramene a literatúru, z ktorých som čerpal.

V Žiline, dňa 29. apríla 2013 Podpis .....

Poď akovanie

Chcel by som sa poďakovať všetkým osobám, ktoré sa priamo alebo nepriamo zapojili do tvorby tohto diplomového projektu a pomohli, či už radou, kontrolou alebo pozitívnou motiváciou.

Ďakujem taktiež všetkým členom tímu - či už kolegom študentom, ako aj učiteľom, ktorí sa nám venovali.

Najväčšia vďaka patrí vedúcej diplomovej práce, pani Ing. Kataríne Zábovskej, PhD. Jej neúnavná snaha asistovať nám v problémových oblastiach, motivovať nás a združovať ako tím nám veľmi pomohla pri dokončení našej práce.

Veľká vďaka patrí takisto rodine a kamarátom, ktorí mi boli v čase tvorby práce na blízku a podali pomocnú ruku.

Obsah

Zoznam obrázkov

Zoznam použitých skratiek

AIS	Advanced Integration Service
API	Application Programming Interface
BRMS	Business Rule Management System - Podnikový systém na riadenie pravidiel
CSV	Comma separated values
DDL	Data Definition Language
DIČ	Daňové Identifikačné číslo
DML	Data Manipulation Language
EBITDA	Earnings Before Interest, Taxes, Depreciation and Amortization
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ESQL	Extended standard query language
ETL	Extract, transform, load
FTP	File Transfer Protocol
HTTP	Hyper text transfer protocol
HTTPS	Hyper text transfer protocol secure
IČO	Identifikačné Číslo Organizácie
IDE	Integrated Development Environment
JMS	Message Service
JVM	Java Virtual Machine
KPI	Key Performance Indicator - Kľúčový indikátor výkonu
MOM	Message Oriented Middleware
MQ	Message Queue
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
PHP	Post Hypertext Preprocessor
REST	Representational State Transfer
RPC	Remote Procedure Call - Vzdialené volanie procedúr
RUP	Rational unified process
SCA	Service Component Architecture - SCA

SFTP	Secure File Transfer Protocol
SOA	Service oriented architecture
SOAP	Simple Object Access Protocol
SOI	Service Oriented Integration
SQL	Standard query language
TCP/IP	Transmission Control Protocol / Internet Protocol
WTX	WebSphere Transformation Extender
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet language

Úvod

Moja diplomová práca je zameraná na integráciu služieb prostredníctvom ESB (enterprise service bus) podľa SOA (service oriented architecture) princípov na IBM WebSphere Message Broker platforme. Práca je súčasťou tímového súťažného projektu pre spoločnosť IBM, do ktorého sa zapojila aj Fakulta riadenia a informatiky.

Úlohou tímu bolo vyvinúť systém monitoringu portfólia bankových klientov. V rámci plnenia tohto cieľa nám boli pridelené konkrétne úlohy a časti aplikácie, avšak výsledný systém je dielom práce tímu ako celku.

Praktická časť diplomovej práce spočívala v analýze problémovej oblasti (banky a úvery), teoretickému a praktickému zvládnutiu nástroja IBM WebSphere Message Broker a návrhu, implementácie a testu EWS (systému včasného varovania).

V prvej kapitole sa zaoberám všeobecnými princípmi integrácie systémov, podľa ktorých je navrhnutá aj architektúra EWS. Dôraz sa kladie najmä na trojvrstvovú architektúru, SOA, ESB, referenčné architektúry a databázové systémy.

Druhá kapitola opisuje technológie, ktoré sa využívajú na tvorbu EWS. Jedná sa o produkty WebSphere. BPM a ILOG sú rozobraté stručnejšie a ich popis je predmetom diplomovej práce ďalších študentov, Message Broker je spracovaný podrobne, pretože veľká časť mojej práce bola implementovaná v tomto nástroji.

Tretia kapitola analyzuje samotnú banku, jej fungovanie, princípy a možnosti kontroly a akcií pri defaulte klienta.

Štvrtá kapitola popisuje nami navrhovaný systém od jeho konceptov, cez dátový model až po konkrétne navrhnuté procesy a javy. Návrh vychádza z analýzy banky, je navrhnutý podľa architektúr a štandardov popísaných v teórií v prvej kapitole a prakticky implementovaný podľa

postupov popísaných v kapitole druhej.
Piata kapitola sa venuje konkrétnej podobe EWS, popisuje užívateľské rozhranie, testy, ktoré je možné na systéme vykonať a uvádza ekonomické zhodnotenie celého riešenia.
Keďže boli úlohy na projekte rozdelené v rámci tímu, práca na platforme WebSphere Message Broker sa stala úlohou mňa a Juraja Branického. Zodpovednosti boli podľa zadania zadefinované nasledovne:

- Dátový model bol rozdelený na komponenty, kde každý z nás má konkrétne navrhnúť inú časť, viď kapitola 4.2.
- Implementácia servisov bola rozdelená na niekoľko častí, presnejší popis uvádzam v kapitole 4.5.
- Príprava najmä transakčných dát jednotlivých klientov bola zodpovednosťou Juraja Branického, zatiaľ čo ja som viac skúmal možnosti testov a navrhoval som komponentný model.

Samotný návrh modelu a servisov, práca v nástroji, ktorý nebol predmetom výučby v rámci štúdia či EWS ako celok sú už tímovou prácou. Popísané skutočnosti v tejto práci sú mojím vlastným dielom a pohľadom na celú problematiku.
Osobným prínosom práce pre mňa bola praktická realizácia teoretických štandardov a techník, práca v nástroji Message Broker a práca v tíme. Novou skúsenosťou bola práca s komplexnými platformami, ktoré ponúkajú veľké množstvo funkcionality a je náročné určiť ich úlohu vo finálnom systéme. Prínosom EWS je zlepšenie práce s rizikom banky, väčší zisk, efektivita a flexibilita.
Jednotlivé skratky používané v práci sú vysvetlené v slovníku pojmov. Anglické výrazy boli prekladané len v prípade, že majú ekvivalent v Slovenskom jazyku.

Kapitola 1

Princípy integrácie systémov

S nárastom veľkosti a komplexnosti jednotlivých aplikácií prichádza prirodzene i potreba definovania osvedčených a správnych princípov na ich tvorbu. Tieto princípy sú efektívne na základe racionálnych a dokázateľných faktov, ako aj empirických skúseností.
Aplikácia systematického, disciplinovaného a kvantitatívneho prístupu k vývoju, prevádzke a údržbe softvéru sa nazýva softvérové inžinierstvo (software engineering). Z vyššej úrovne sa návrhom a štruktúrou softvérových systémov zaoberá softvérová architektúra (software architecture).
Riadenie vývoja ďalej určujú rôzne metodiky, ktoré popisujú postupy na samotný vývoj softvéru a rozdeľujú ho zvyčajne do niekoľkých krokov.
Dôvodov na tvorbu takýchto teórií, počnúc metodikami a končiac softvérovým inžinierstvom, je veľké množstvo, medzi nimi aj:

- v softvéri sa vyskytuje veľké množstvo chýb (bugov),
- softvér nie je vyvíjaný na čas,
- softvér je náročné udržiavať,
- softvér sa ťažko prispôsobuje zmenám,
- softvér nefunguje správne (je pomalý, neefektívny, nestabilný atď).

Pri dizajnovaní komplexnej podnikovej aplikácie môžeme vopred predpokladať väčšie množstvo komponentov, vyžitie údajových štruktúr i databáz na ukladanie dát spolu s určitou obchodnou logikou, kvôli ktorej je aplikácia vlastne v prvom rade vyvíjaná.
Ak poznáme jednotlivé komponenty, z ktorých sa bude systém skladať a vieme ich vzťahy, je možné zvoliť systémovú architektúru. Architektúra je navrhnutá podľa určitých vzorov, ktoré uľahčujú jej chápanie. Porozumenie architektúre na nižšej úrovni môže byť veľmi náročné (napríklad zo zdrojových kódov), no pri použití vzorov sa stáva jej štruktúra zrejmou.
Vzorov je väčšie množstvo, avšak pre potreby mojej diplomovej práce uvediem najmä n-vrstvový model a jeho konkrétnu implementáciu v podobe 3-vrstvového modelu. Inými vzormi sú napríklad tabuľa, rúry a filtre, mikrojadro a iné.

1.1 Trojvrstvová architektúra

Vrstvy predstavujú spôsob ako rozložiť systém na najvyššej úrovni. Rozdelenie a zlučovanie jednotlivých komponentov v systéme sa vykonáva na základe ich zodpovednosti, komponenty s rovnakými zodpovednosťami sa zlučujú do jednej vrstvy.
Každá vrstva je tak zložená z niekoľkých komponentov, ktoré by mali mať spoločné určité vlastnosti, minimálne približne rovnakú úroveň abstrakcie.
Jedným z najznámejších typov viacvrstvovej architektúry je trojvrstvová architektúra. Jednotlivé vrstvy sú:

- **Prezentačná vrstva** - poskytuje aplikačné (grafické) užívateľské rozhranie (user interface - UI, často tiež graphical user interface - GUI).
- **Obchodná vrstva** - implementuje obchodnú logiku a funkcionalitu aplikácie.
- **Dátová vrstva** - poskytuje prístup k dátam - častokrát uloženým v externých systémoch - databázach.

Alternatívne môžeme jednotlivé vrstvy nazvať aj front end (prezentačná vrstva), obchodná logika (obchodná vrstva) a back end (dátová vrstva).

1.1.1 Front end / portálové riešenie

Ak je obchodná logika aplikácie sama o sebe zložitá, ak využíva veľké databázy a vykonáva nad nimi komplexné dotazy, tak s vysokou pravdepodobnosťou bude musieť byť aj užívateľské rozhranie podobne zložité.
Existuje mnoho teórií o tvorbe správneho užívateľského rozhrania. Jeho štruktúra závisí od platformy, prostredia či spôsobu ovládania nástroja, na ktorom bude softvér bežať (telefón s dotykovým displejom vs. počítač).
V závislosti od použitia systému je častokrát front end tvorený so snahou zaujať zákazníka grafickým prevedením, jednoduchým ovládaním a prístupom k informáciám (napríklad webové stránky). Inokedy je cieľom najmä poskytnúť úplnú kontrolu nad obchodnou logikou s veľkým množstvom informácií na podporu rozhodnutia, keď je snaha o upútanie náhodného užívateľa minimálna.
Súčasťou frontendu sú rôzne formuláre, na ktorých sa vypĺňajú dáta. Ich štruktúra závisí od požiadaviek obchodných procesov, ktoré potom operujú s dátovou vrstvou. Ideálnym riešením je prevolávanie SOA servisov, viac o SOA architektúre uvádzam v sekcií 1.2.

1.1.2 Obchodná logika

Aplikácia akéhokoľvek rozsahu môže byť štruktúrovaná na koncepte obchodných procesov a obchodných komponentov. Táto „obchodná logika“ je jasne oddelená od prezentačnej či dátovej vrstvy, môže však využívať služby z iných vrstiev a tiež im poskytuje svoje služby cez rozhrania. V trojvrstvovej architektúre funguje takáto komunikácia vždy jednosmerne. To znamená, že dátová vrstva poskytuje služby obchodnej vrstve a tá zase prezentačnej vrstve. Prezentačná vrstva by nemala priamo využívať služby dátovej vrstvy a obchodná vrstva by nemala implementovať služby pre dátovú vrstvu.
Peter Herzum a Oliver Sims vo svojom diele [2] definujú obchodnú vrstvu ako: „Softvérová implementácia autonómnych obchodných konceptov alebo obchodných procesov. Skladá sa zo všetkých softvérových artefaktov, ktoré sú potrebné na reprezentáciu, implementáciu a nasadenie daných obchodných konceptov, ako automómne, znovu-použiteľné elementy vo väčšom, distribuovanom informačnom systéme.“

Pod pojmom obchodný komponent rozumieme softvérovú realizáciu obchodného konceptu. Zaobaľuje v sebe obchodnú logiku, taktiež nazývanú obchodné pravidlá. Tieto pravidlá určitým spôsobom ohraničujú správanie jednotlivých procesov a sú špecifické pre danú spoločnosť, pre ktorú je informačný systém vyvíjaný.
Príkladom na obchodnú logiku v bankovom prostredí je udeľovanie úveru zákazníkovi. Proces ako taký vyžaduje informácie o klientovi, ktorého žiadosť má posudzovať a jeho výsledkom je odpoveď áno/nie, inak povedané, či mu bude úver udelený, respektíve udelený nebude. Proces však môže byť vnútorne veľmi komplikovaný a spravidla si vyžadovať mnoho dát z dátových štruktúr prípadne databáz, či zložitých prepočtov. Cieľom je, aby túto logiku zapuzdroval v sebe a jasne definoval, aké dáta na svoje beh potrebuje a aké výsledky môže vrátiť. Jednou z možností na takéto zapuzdrenie logiky a štandardizované definovanie prístupu je aj servisne orientovaná architektúra - SOA. O servisoch a ich vlastnostiach uvádzam viac v ďalších kapitolách.

1.1.3 Back end / dátová vrstva

Obchodné aplikácie potrebujú prístup k dátam uložených v rôznych formátoch. Najpoužívanejším spôsobom je v dnešnej dobe databáza, ako množina permanentne uložených n-tíc dát. Najčastejšie sa využívajú relačné databázy, alternatívu tvoria napríklad objektové databázy. Vo väčších spoločnostiach s veľkým množstvom dát, pri ktorých je dôležité sledovať ich zmeny v čase, je výhodné vytvoriť dátový sklad. Okrem databáz je možnosť ukladať dáta do súborov (binárnych alebo textových) alebo ich držať iba v operačnej pamäti.
Samotná dátová vrstva je zodpovedná za poskytovanie takto uložených dát obchodnej vrstve. Jednou z možností takéhoto poskytovania dát sú servisy. Cieľom je ponechať rovnakú signatúru jednotlivých servisov a skryť tak ostatné detaily ich vykonávania.
Dalo by sa povedať, že dátová vrstva izoluje obchodnú vrstvu od špecifikácií prístupu k dátam. Prináša to so sebou nasledujúce výhody:

- dopad zmeny databázy je minimálny,
- dopad zmeny architektúry databázy je minimálny,

- dopad zmien v schémach, samotných tabuľkách či riadkoch je minimálny,
- v servisoch je zapúzdrený všetok kód slúžiaci na manipuláciu s dátami - uľahčuje sa tak jeho údržba a testovanie.

Minimalizácia dopadu zmien je možná vďaka jednoduchému faktu. Ak zabalíme jednotlivé požiadavky obchodnej vrstvy do servisov, tak meníme servis iba raz, na mieste, kde je definovaný, bez toho, aby sme vôbec potrebovali vedieť, kde presne je servis volaný.
V praxi by samotný servis mohol vyžadovať jednoduché parametre typu ID konkrétneho objektu, o ktorom chcem zistiť viac informácií alebo napríklad odsadenie a počet vrátených prvkov. Vrátené dáta budú v predom definovanej údajovej štruktúre alebo v objekte, ktorý poznajú obe vrstvy.
Pri volaní servisu nemusíme vedieť, akú konkrétnu databázu či iné úložisko používame, ani jej verziu, schému, dátový model, nie je nutné sa dotazovať na dáta prostredníctvom jazyka SQL - standard query language - alebo jednou z jeho modifikácií.
Ak by došlo k zmene štruktúry databázy (pri jej optimalizácií, prípadne pridávaní ďalších entít), tak by sa menilo samotné telo servisu - SQL príkaz, mapovanie dát na objekt a podobne. Takúto zmenu by však bolo možné vykonať bez nutnosti zmien či zásahov v obchodnej vrstve.
Jednou z možností na zapuzdrenie sú opäť SOA servisy. Alternatívou vo väčšine programovacích jazykoch sú triedy, ktoré prácu s konkrétnymi databázami zaobaľujú. Tie však samy o sebe netvoria dátovú vrstvu, tá by mala poskytovať už konkrétne metódy na získanie požadovaných dát. Pre komunikáciu na podobnej úrovni ako umožňujú SOA servisy by tak bolo nutné urobiť ešte niekoľko krokov, ktoré však nie sú pre túto prácu potrebné.
Príkladom na využitie dátovej vrstvy v praxi je servis na získanie informácií o klientovi. Klient môže byť jednoznačne identifikovaný napríklad rodným číslo a číslom občianskeho preukazu. Po ich zadaní sa v dátovej vrstve vykonajú príslušné databázové dotazy, pospájajú sa údaje z niekoľkých tabuliek, schém či databáz a vrátia sa obchodnej vrstve, v dohodnutej podobe. Alternatívne, ak by sme vyhľadávali napríklad podľa priezviska, servis by mohol vracať zoznam osôb s daným alebo podobným priezviskom usporiadaným podľa relevancie a pri zvolení jedného z nich by sa prevolal servis na základe identifikátora zvoleného klienta.
Presnejšie je dátová vrstva popísaná v kapitole 1.4.

1.2 SOA - servisne orientovaná architektúra

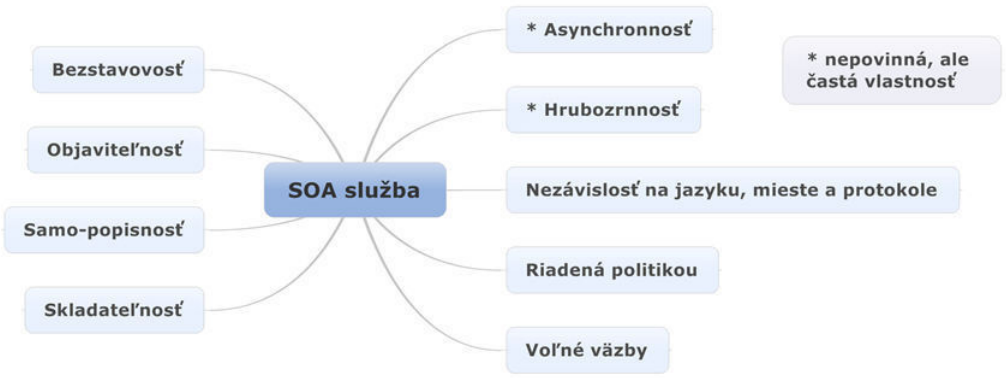
Definícia SOA

Service oriented architecture (servisne orientovaná architektúra - ďalej len SOA) patrí medzi často skloňované a používané pojmy v súčasnosti. Takmer vždy sa vyskytuje v sloganoch moderných spoločností či pri popisoch efektívnych prístupov. Čo však SOA v skutočnosti je? Ak by sme sa spýtali dvoch rôznych osôb, pracujúcich SOA, čo tento pojem vlastne znamená, dočkali by sme sa pravdepodobne dvoch rôznych definícií, ktoré by však mohli byť správne. Podľa zdroja [11] sa jedná o: „Voľne zviazanú architektúru, dizajnovanú tak, aby vyhovovala obchodným požiadavkám organizácie.“
Voľná väzba v architektúre (po anglicky loose coupling) znamená, že obe strany v takomto spojení majú len malú vedomosť o tom, ako funguje tá druhá strana, vedia však, aký výsledok od nej môžu očakávať.
SOA je prístup k návrhu, implementácií a riadeniu distribuovaného spracovania, ktoré daná firma potrebuje k realizácií svojej stratégie a k dosiahnutiu obchodných cieľov. SOA je množina servisov, ktoré sú dostupné zákazníkom, partnerom a iným častiam organizácie. Tieto servisy sa zavádzajú z viacerých dôvodov, medzi ktoré patrí:

- snaha o zníženie nákladov (služby je možné používať opakovanie, efektívnejší vývoj a údržba),
- snaha o zníženie risku,
- snaha o zníženie času na implementáciu,
- zvýšenie tzv. agility (využívajúc agilné metódy, t.z. flexibilnejšie a rýchlejšie metódy),
- lepšia kontrola nad procesmi a ich transparentnosť,
- inkrementálna implementácia.

Vlastnosti SOA servisov

Obr. 1.1: Grafické znázornenie vlastností SOA servisov



Servis v SOA architektúre je logická obchodná funkcia a má nasledovné vlastnosti:

Bezstavovosť' SOA servisy si „nepamätajú" požiadavku, ktorá predchádzala tej súčasnej a nevedia ani, aká požiadavka bude nasledovať. Nezávisia tak ani na kontexte, ani na stave iných servisov, ale iba na ich funkcionalite. Napríklad, telefonát je stavový dej, písanie listu je bez-stavové.

Objavitelnosť' Servis musí byť objaviteľný jeho potenciálnymi odberateľmi. Platí jednoduché pravidlo, ak nikto nevie, že servis existuje, je malá šanca, že bude používaný.

Samo-popisnosť' Rozhranie SOA servisu popisuje a poskytuje vstupný bod pre tento servis. Podstatné je, že takéto rozhranie poskytne potenciálnemu používateľovi servisu všetky informácie na jeho používanie, bez toho, aby tento používateľ rozumel alebo poznal technické detaily implementácie.

Skladateľnosť' Servisy sú od základu skladateľné - na implementáciu jedného servisu sa môžu využiť iné servisy, takže vo všeobecnosti môžu byť kombinovateľné za účelom vytvorenia nového obchodného riešenia.

Voľná väzba Princíp voľnej väzby / viazanosti umožňuje rozdeliť aplikačné záležitosti do nezávislých častí. Dosahuje sa to stanovením hraníc, ktoré môžu byť fyzické alebo logické a ležia medzi jednotlivými sadami úloh.

Riadená politikou Vzťahy medzi servismi sú riadené politikou a dohodami na úrovni servisov (SLA - service-level agreement), aby sa pozdvihla úroveň konzistencie a znížila ich zložitosť.

Nezávislosť na mieste, jazyku či protokole Servisy by mali byť vo všeobecnosti prístupné ktorýmkoľvek autorizovaným používateľom, na akejkoľvek platforme a z akejkoľvek oblasti.

Okrem toho majú servisy nasledovné charakteristiky:

Hrubozrnnosť Zrnitosť sa v tomto kontexte vzťahuje na bohatosť funkcií ponúkaných servisom (ktorý je zvyčajne obchodným servisom). Ak je servis hrubozrnný, na vykonanie určitého úkonu je potrebných menej krokov (pre vývojára systému). Aj aplikácie sú prirodzene hrubozrnné, pretože v sebe zaobaľujú širokú funkcionalitu. Príkladom hrubozrnného servisu môže byť získanie informácií o účte (pretože v sebe zahŕňa návrat mena, čísla účtu a napríklad adresy). Samotný servis na získanie čísla účtu by bol potom jemnozrnný.

Asynchrónnosť Asynchrónnosť nepatrí medzi nevyhnutné požiadavky na servisy v SOA architektúre, ale umožňuje obísť problémy s pomalou odozvou siete alebo veľkými komunikačnými nákladmi. Asynchrónne správanie v praxi znamená, že sa vykoná požiadavka na servis a potom program pokračuje ďalej bežným spôsobom, až kým príde odpoveď na požadovaný servis.

1.2.1 SOA integrácia

Integrácia s použitím SOA princípov je v súčasnosti veľmi aktuálny pojem, ktorého význam sa posúva a pretvára. Je súčasťou systémovej integrácie, ktorej cieľom je previazať jednotlivé individuálne softvérové aplikácie a systematicky ich riadiť. Okrem toho sa tento pojem rozširuje i do oblasti vízií spoločnosti a zaistenia jej konkurencieschopnosti.
Úlohy systémovej integrácie by sa dali zhrnúť do nasledovných bodov:

- Zlúčenie riadenia organizácií a ich informatiky.
- Integrácia informačného systému s horizontálnymi alebo vertikálnymi podnikovými procesmi.
- Riadiť podníkovú informatiku.
- Integrácia podnikovej informatiky s obchodnými a inými podnikovými aktivitami.
- Integrovať podnik s okolím (e-business, sociálne siete a iné).
- Integrácia vízií - zapojenie podnikovej informatiky do riadenia firmy ako celku.

Tieto ciele a najnovšie trendy boli spracované podľa prieskumu uverejneného v zdroji [7].
Keďže rastie aj zložitosť jednotlivých systémov, začínajú sa zavádzať služby, ktoré si každý podnik kategorizuje podľa svojich potrieb.
V tomto kroku začína byť pre systémovú integráciu zaujímavá práve SOA architektúra, ktorú som popísal v predchádzajúcej kapitole. SOA definuje služby (alebo aj servisy), ktoré sa využívajú v rámci organizácie. Samotná SOA integrácia sa nazýva SOI (service oriented integration). Servisy sú vystavené zbernici, nie konkrétnemu odberateľovi (viď voľná zviazanosť).
Z implementačného hľadiska sú správy medzi servismi posielané pomocou SOAP správ. SOAP je protokol, ktorý definuje ich tvar a skladá sa z obálky, ktorá obsahuje špecifickú hlavičku a samotnú správu. Alternatívou k SOAP je napríklad protokol REST.
Servis v sebe môže zaobaľovať niekoľko operácií. Ak servis rieši obsiahlejší problém, ktorý je súčasťou obchodnej logiky organizácie, nazýva sa obchodným servisom. Príkladom obchodného servisu môže byť schválenie požiadavky na úver, ktoré v sebe zaobaľuje celú radu

zložitejších operácií. Takýto postup sa nazýva aj **orchestrácia servisov**.
Orchestrácia servisov je koordinácia a využitie viacerých servisov, ktoré sú navonok prístupné ako jeden agregovaný servis. Tým sa využívajú ich vyššie spomínané vlastnosti. V architektúre vyvíjaného systému včasného varovania je SOA integrácia jednotlivých súčastí kľúčovým prvkom.

1.2.2 Integračné štandardy

Systémová integrácia, ktorej zmysel bol popísaný v kapitole 1.2.1, sa riadi určitými štandardami a existuje niekoľko zaužívaných základných metód na jej realizáciu:

- **Vertikálna integrácia** (ako protiklad horizontálnej integrácie) integruje podsystémy podľa ich funkcionality. Táto metóda sa vo všeobecnosti považuje za rýchlejšiu a lacnejšiu (z krátkodobého hľadiska). Nevýhodou je slabá prispôsobivosť zmenám vo funkcionalite, keď je zložité upravovať už existujúce informačné „silá“.

- **Integrácia do hviezdy** (tiež nazývaná **špagetová integrácia**) je integračná metóda, v ktorej je každý podsystém pripojený ku všetkým ostatným podsystémom. Z pohľadu jedného podsystému pripomína diagram takejto metódy hviezdu, avšak pri celkovom náhľade sa ujal názov špagety, vzhľadom na tvar diagramu. Výhodou je extrémna flexibilita a znovu-využívanie funkcionality, avšak s každým novým podsystémom exponenciálne narastajú náklady.

- **Horizontálna integrácia** (alebo aj **enterprise service bus**) je založená na špecializovanom podsystéme, ktorý má za úlohu zabezpečiť komunikáciu medzi ostatnými podsystémami. Viac informácií uvádzam v kapitole 1.3.

- **Spoločný formát** dát je integračná metóda, pri ktorej sa poskytuje servis na konverziu dát z formátu, ktorý je špecifický pre aplikáciu, na bežné dátové formáty. Nie je tak nutné, aby každý podsystém konvertoval dáta sám, pomocou špecifických adaptérov.

1.2.3 Referenčné architektúry

Referenčná architektúra poskytuje v oblasti podnikových či softvérových architektúr vzorové riešenie v určitej doméne. Okrem toho definuje aj určitý základný spoločný „slovník“, pomocou ktorého sa dá opísať následná implementácia.

Referenčná architektúra sa definuje na rozdielnych úrovniach abstrakcie. Pre dostatočnú flexibilitu sa úrovne abstrakcie principiálne podobajú na pohľady v rámci RUP (rational unified process). Jedná sa o logický, implementačný, procesný a nasadzovací pohľad + pohľad prípadov použitia. Takéto rozdelenie však nie je povinné a existujú aj alternatívne prístupy.

Použitie referenčnej architektúry vo firme vyžaduje určitý tím ľudí, ktorý dokáže určiť preferované spôsoby a vzorové riešenia problémov. Tieto rozhodnutia musia byť zdokumentované a správne podložené. Počas fungovania spoločnosti, ktorá využíva referenčné architektúry sa môže stať, že je potrebné túto architektúru aktualizovať, prípadne upraviť.

V praxi by to znamenalo vykonanie dodatočných rozhodnutí - ak je napríklad potrebné využívať web servisy a referenčná architektúra vôbec neurčuje ich technológiu. Vtedy je nutné rozhodnúť, či sa bude využívať SOAP alebo vzdialené volanie procedúr - RPC, alebo v akom jazyku a s využitím akého štandardu sa budú servisy implementovať.

Hlavná výhoda referenčnej architektúry je teda šetrenie času, efektivita a v konečnom dôsledku lepšie fungujúca spoločnosť. Odpadáva tak nutnosť vykonávať analýzu vždy, ale naopak, využiť už zistené informácie, a tak sa pri implementácií vybrať ďalej tou správnou cestou.

Snahou referenčných architektúr rozhodne nie je obmedzenie kreativity, ale skôr zaistenie zhody medzi jednotlivými projektami. Vertikálne znovu využívanie informácií medzi projektami je už dnes medzi spoločnosťami bežné, avšak to horizontálne je zriedkavé, najmä pri stredne veľkých spoločnostiach. Práve v tomto spočíva význam a potenciál referenčných architektúr.

Informácie o praktickom využití referenčných architektúr som čerpal z [13].

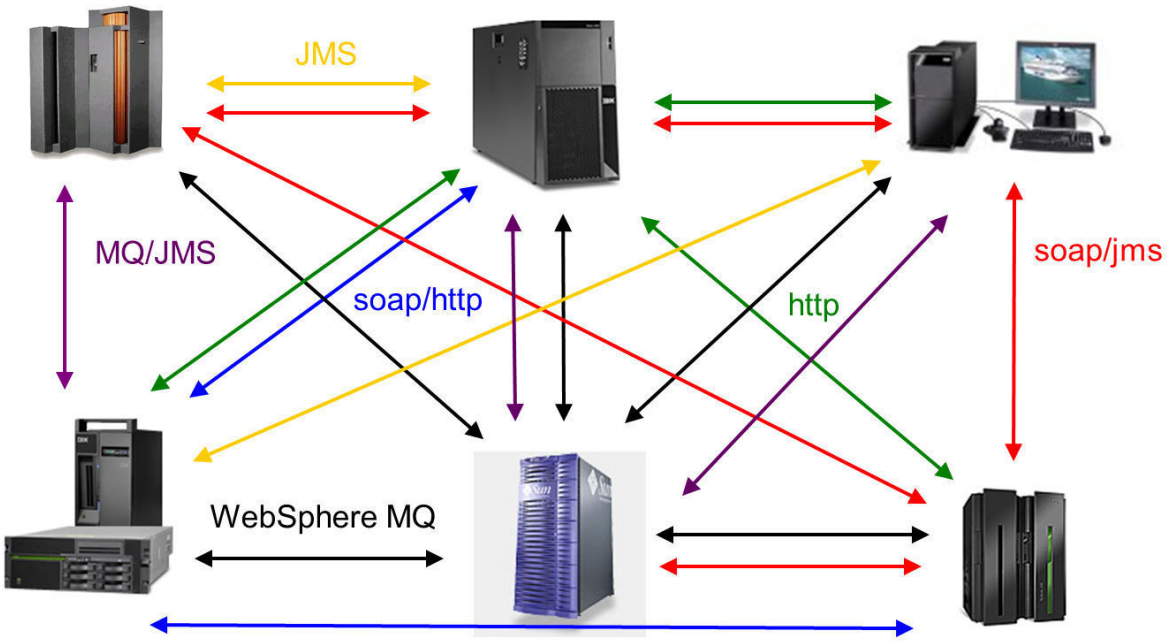
1.3 ESB - Enterprise Service Bus

Enterprise Service Bus (podniková servisná zbernica - ďalej len ESB) je model softvérovej architektúry pre integráciu aplikácií a servisov, inak povedané pre **systémovú integráciu**.
ESB prakticky vykonáva tieto 4 operácie:

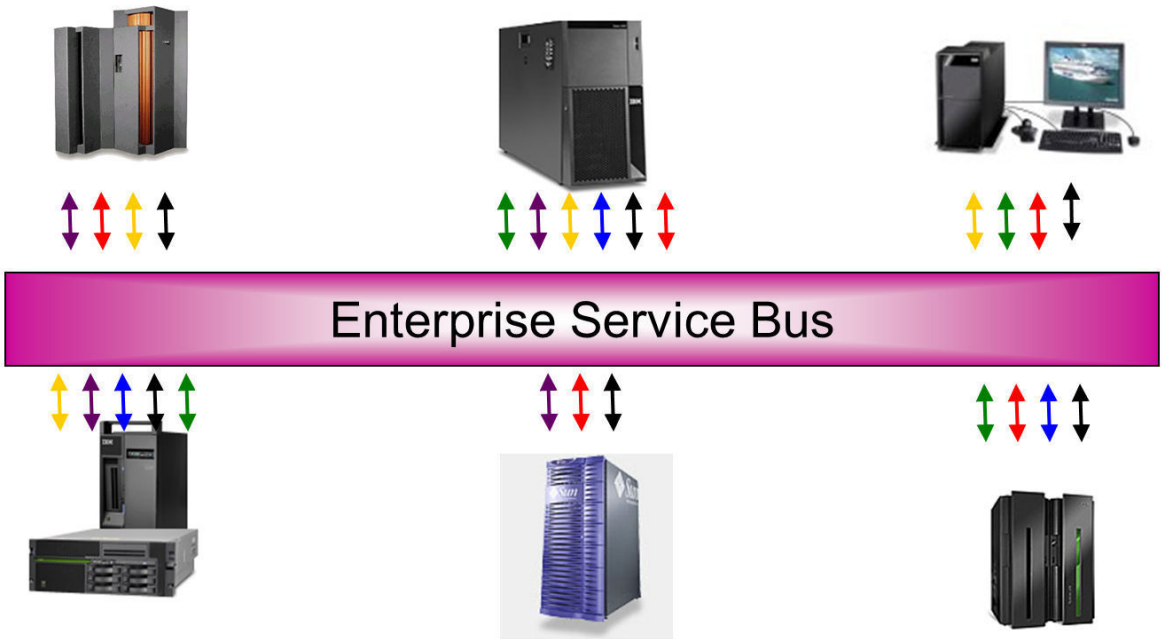
1. **Spája a smeruje** komunikáciu medzi servismi.
2. **Vykonáva konverziu** medzi rôznymi transportnými protokolmi.
3. **Transformuje** rôzne dátové formáty.
4. **Identifikuje a distribuuje** obchodné udalosti.

Výhodou ESB je jasné „oddelenie záujmov“. V tomto prípade sa jedná o oddelenie aplikácií, ktoré majú obchodné funkcie (obchodné servisy a obchodné procesy) a ESB (infraštruktúry pre spájanie aplikácií a servisov dohromady).
Cieľom tohto oddelenia a zvýšenej flexibility je vyťažiť maximum z vlastností a hodnôt SOA. Dosiahne sa tak, že každá IT zmena má obmedzený dosah na celkové fungovanie systému. Zmien môže byť veľké množstvo - napríklad aktualizácia už existujúcich servisov, ich nahradenie novými obchodnými servismi alebo vytvorenie nového, kompozitného riešenia, ktoré vplýva na ostatné servisy.
V tomto prípade sa teda „zodpovednosť“ za komunikáciu s ostatnými podsystémami presúva na 1 podsystém. Je teda jednoduchšie a lacnejšie nahrádzať už existujúce podsystémy a nie je nutné hľadať dopady takýchto zmien vo všetkých ostatných podsystémoch.
Je možné zameniť jeden podsystém za druhý, poskytujúci podobnú funkcionalitu, ale iné komunikačné rozhrania, bez vplyvu na ostatné podsystémy. Jedinou nutnou akciou je implementovať nové rozhranie medzi ESB a pridávaným podsystémom.

Obr. 1.2: Podnik: Situácia pred použitím ESB



Obr. 1.3: Podnik: Situácia po použití ESB



1.4 Databázové systémy

1.4.1 Definícia a rozdelenie

Databázový systém umožňuje efektívne ukladanie a správu dát. Skladá sa z databázy (perzistentná množina štruktúrovaných dát) a systému riadenia bázy dát (v skratke SRBD). SRBD je podľa zdroja [8] definovaný ako: „Systém riadenia bázy dát je množina programov, zabezpečujúcich manipuláciu s dátami, ochranu dát, paralelné spracovanie, apod.“.
Na vytváranie a definíciu dátových štruktúr, schém či tabuľkových priestorov, vytvorenie a úpravu samotných tabuliek sa využíva DDL, data definition language.
Operácie nad dátami sú definované pomocou DML - data manipulation language. Najvyužívanejším jazykom na manipuláciu s dátami je SQL - standard query language. Nad dátami sú definované 4 základné operácie:

- INSERT - vkladanie nových dát,
- UPDATE - úprava existujúcich dát,
- DELETE - mazanie dát,
- SELECT - výber dát.

Dáta sú organizovné už do spomínaných tabuliek, tabuľky do tabuľkových priestorov a tie do schém, ktoré prislúchajú databázam. Dáta sa dajú rozdeliť na tieto kategórie:

- Vstupné dáta - ktoré do systému prichádzajú, sú vkladané.
- Perzistentné dáta - ktoré v systéme zostávajú trvalo.
- Výstupné dáta - ktoré prezentujú výstupné informácie zo systému.

Databázový systém môže byť navrhnutý podľa rôznych architektúr, ktoré vo všeobecnosti vysvetľujú jeho štruktúru. Architektúra je rozdelená na 3 základné úrovne - schémy (množiny dát, ktoré popisujú dátový model):

- Interná schéma - implementačne závislá množina dát, fyzická štruktúra databázy, kompletné detaily umiestnenia dát.

- Konceptuálna schéma - implementačne nezávislá množina dát, ktorá popisuje štruktúru databázy pre určitú skupinu používateľov, zakrýva detaily fyzickej štruktúry. Zahŕňa popis entít, ich atribútov a vzťahov medzi entitami.
- Externá schéma - implementačne nezávislá množina dát, popisuje používateľské pohľady, časti databázy pre vybranú skupinu užívateľov, ignorovanie nepodstatných detailov.

Databázové systémy majú mnoho vlastností, ktoré vyplývajú z uvedených definícií. Popis týchto vlastností však nie je pre potreby tejto diplomovej práce nutný.
Z historického hľadiska vzniklo niekoľko základných prístupov k tvorbe a návrhu databáz, od hierarchických a sieťových databáz, cez relačné až po objektové databázy. V rámci navrhovaného systému včasného varovania sa využívajú v súčasnosti najrozšírenejšie relačné databázy. Medzi známe a rozšírené systémy riadenia bázy patria:

- SQLite - voľne dostupný SRBD, využívaný napr. aplikáciami v mobilnom operačnom systéme Android.
- MySQL - voľne dostupný SRBD, často využívaný na menších webových stránkach.
- Microsoft SQL Server - proprietárny SRBD vyvíjaný spoločnosťou Microsoft.
- Oracle Database - proprietárny SRBD vyvíjaný spoločnosťou Oracle, podľa prieskumov najrozšírenejší.
- IBM Informix - SRBD odkúpený spoločnosťou IBM, pokračuje sa v jeho vývoji a modernizácií.
- IBM DB2 - SRBD vyvíjaný spoločnosťou IBM, využívaný aj v systéme včasného varovania.
- SyBase - SRBD vyvíjaný spoločnosťou SyBase Inc.

Z uvedených SRBD sa v rámci tejto diplomovej práce a vyvíjaného systému včasného varovania bude využívať DB2 od spoločnosti IBM. Dôvodom je jeho dobrá integrácia s nástrojmi z rodiny produktov WebSphere a tiež porovnateľné vlastnosti s ostatnými systémami. Samotné modely sú navrhnuté univerzálne, bez fyzickej naviazanosti na DB2 databázu a jej špecifiká.

1.4.2 Dátové sklady

Spomínané databázové systémy sú založené na tzv. OLTP (online transaction processing) modeli. Využíva sa najmä pri transakčných databázach a chýbajú v ňom komplexné analytické nástroje. Preto je obmedzené aj množstvo dát, ku ktorému je možné pristúpiť v rozumnom čase.

Alternatívou je OLAP (online analytical processing) model, ktorý je zameraný na rýchly prístup k veľkému objemu dát a umožňuje nad nimi vykonávať komplexné analýzy. Je vhodný pre prognózovanie a modelovanie, avšak za cenu značnej väčšej náročnosti na pamäť ako pri OLTP modeli.

Medzi jednotlivými modelmi je niekoľko základných rozdielov, ktoré zároveň určujú ich využitie. OLTP umožňuje krátke a rýchle vkladanie dát v reálnom čase. OLAP naopak dáta vkladá periodicky, pomocou definovaných procesov a z viacerých OLTP databáz. Užívateľ má tak možnosť dáta iba čítať.

Účelom OLTP databáz je kontrola základných činností, OLAP poskytuje pomoc pri plánovaní, podpore rozhodovania a obsahuje historické dáta, na ktoré sa dá dívať cez multidimenzionálne pohľady.

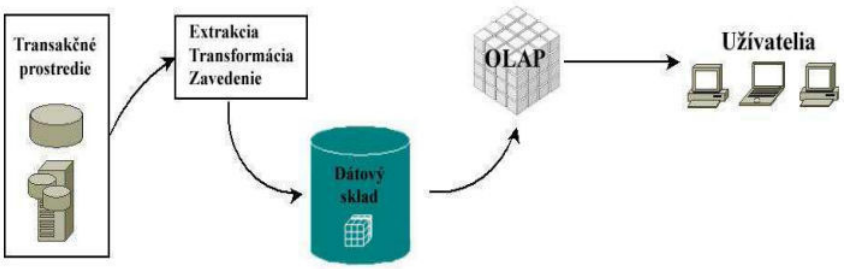
Dáta v OLAP databáze sú zamerané na zodpovedanie zložitejších logických obchodných dotazov a sú štruktúrované. V prípade ich poškodenia je možnosť ich znovu získať z OLTP databáz.

Rozdielov medzi jednotlivými modelmi je ešte viac, ich podrobnejší popis však nie je pre potreby mojej diplomovej práce dôležitý.

Dáta sú usporiadané do tzv. OLAP kocky (tiež aj multidimenzionálna databáza), ktorá je ekvivalentom tabuľky v relačnej databáze. Takáto kocka má niekoľko dimenzií, ktoré obsahujú logicky alebo organizačne usporiadané informácie. S každou ďalšou dimenziou však geometricky narastá aj veľkosť samotnej kocky.

Jednotlivé dimenzie navzájom spája tabuľka faktov. Spolu s tabuľkami dimenzií sa potom spájajú do schém, najznámejšími schémami sú Star schema (hviezdicová schéma) a Snowflake schema (schéma snehovej vločky).

Obr. 1.4: OLAP kocka a dátový sklad



Nad OLAP kockou sa vykonávajú špeciálne operácie. OLAP kocka je určitý pohľad na dátový sklad. Dáta v dátovom sklade majú rôzny pôvod a odlišné usporiadanie od klasických dát v relačných databázach, získavajú sa pomocou ETL procesu. ETL znamená v skratke extract, transform, load (extrakcia, transformácia, načítanie). Správna definícia tohto procesu je najzložitejšou časťou tvorby dátového skladu.

Dátový sklad je v podstate tiež databáza, avšak je riadená a usporiadaná podľa iných pravidiel. Primárne sú dátové sklady vytvárané na podporu rozhodovania. Jednotlivé definície boli spracované podľa zdroja [9].

Bankový sektor je práve odvetvím, kde majú dátové sklady značné využitie a zmysel, aj vrátane nášho systému včasného varovania. Mnohé konkrétne spoločnosti majú svoje dáta organizované v skladoch, z ktorých potrebujeme pre naše účely dáta získavať, preto je ich stručný popis obsiahnutý v mojej diplomovej práci. Ak by v praxi neboli dátové sklady k dispozícií, alebo by ich stav nebol vyhovujúci, získali by sa dáta z pôvodných z drojov, z ktorých čerpajú aj dátové sklady spoločností.

Význam komplexných analýz nad transakciami korporátnych klientov je zjavný. Konkrétne využitie a princípy systému včasného varovania sú popísane v kapitolách 3 a 4.

Kapitola 2

Predstavenie technológií - IBM WebSphere

IBM WebSphere je rodina produktov od spoločnosti IBM, ktoré poskytujú softvér pre SOA prostredia. Tento softvér umožňuje vytváranie dynamických, prepojených a flexibilných obchodných procesov a poskytuje vysoko efektívne aplikačné infraštruktúry pre rôzne obchodné situácie.
IBM WebSphere je aplikačná a integračná softvérová platforma, zahrňujúca servery, servisy a nástroje, ktoré sú potrebné na vytváranie, nasadzovanie, spúšťanie, monitorovanie podnikových webových aplikácií a riešení na rôznych platformách. Definícia bola spracovaná podľa zdroja [14].

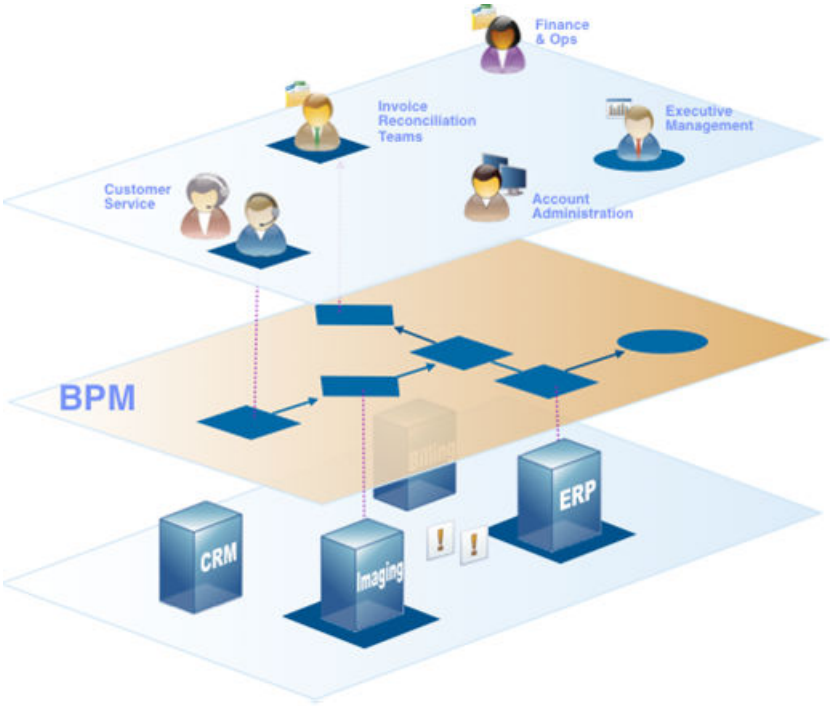
2.1 IBM WebSphere BPM

BPM je skratka pre Business Process Management (po slovensky manažment obchodných / podnikových procesov). BPM je disciplína, ktorá modeluje, riadi, umožňuje automatizáciu, monitoring a optimalizáciu procesov v podniku. Manažment sa vykonáva za účelom zvýšenia efektivity, spokojnosti a prosperity spoločnosti, či za účelom dosiahnutia zisku.
BPM je pre podnik prínosom v mnohých oblastiach, medzi ktoré patrí unifikácia podnikových procesov, využívanie šablón, jednotné užívateľské rozhranie, komplexná správa rolí a

prideľovania úloh, zníženie záťaže ľudských zdrojov, automatizácia, sledovanie a meranie jednotlivých procesov a iné. Podmienkou pre zlepšenie je však dobrý návrh a implementácia jednotlivých procesov, založená na kvalitnej analýze.

BPM vytvára samostatnú vrstvu, ktorá jednotlivé procesy zviditeľňuje a umožňuje nad nimi kontrolu, príklad môžete vidieť na obrázku 2.1.

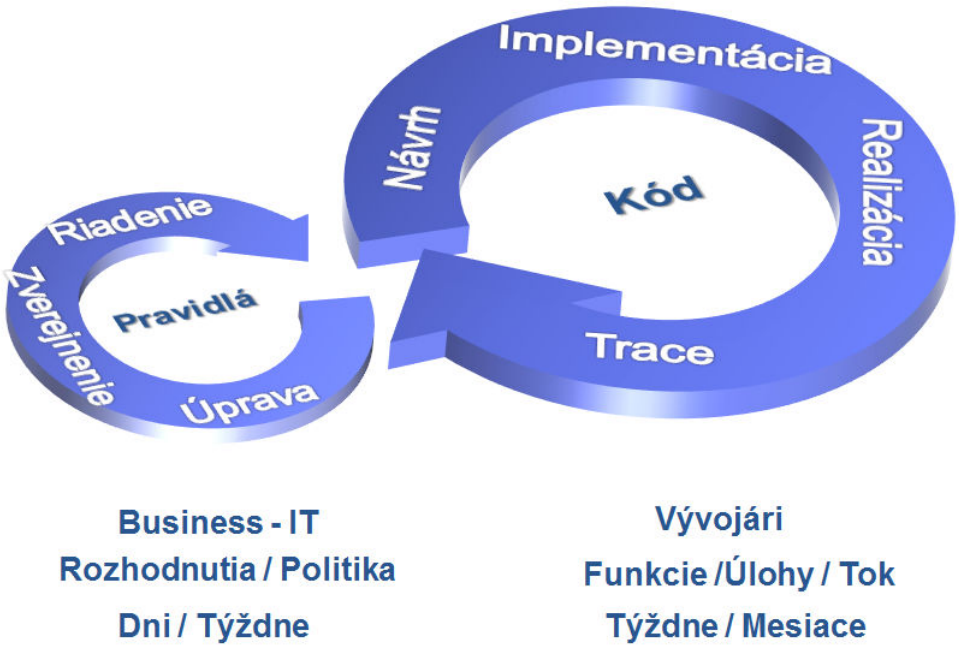
Obr. 2.1: Vrstva BPM



2.2 IBM WebSphere ILOG

IBM Websphere ILOG patrí medzi BRMS systémy, ktoré poskytujú centralizovanú správu pravidiel. Výhodou tohto prístupu je eliminovanie rozptýlenosti obchodných pravidiel (ktoré majú pre procesy zásadný význam) a ich zjednodušený manažment.
Ďalšou veľkou výhodou je oddelenie cyklu zmien pravidiel od celkového cyklu vývoja aplikácie, ako to opisuje aj obrázok 2.2.

Obr. 2.2: Oddelenie vývoja pravidiel od celkového vývoja aplikácie

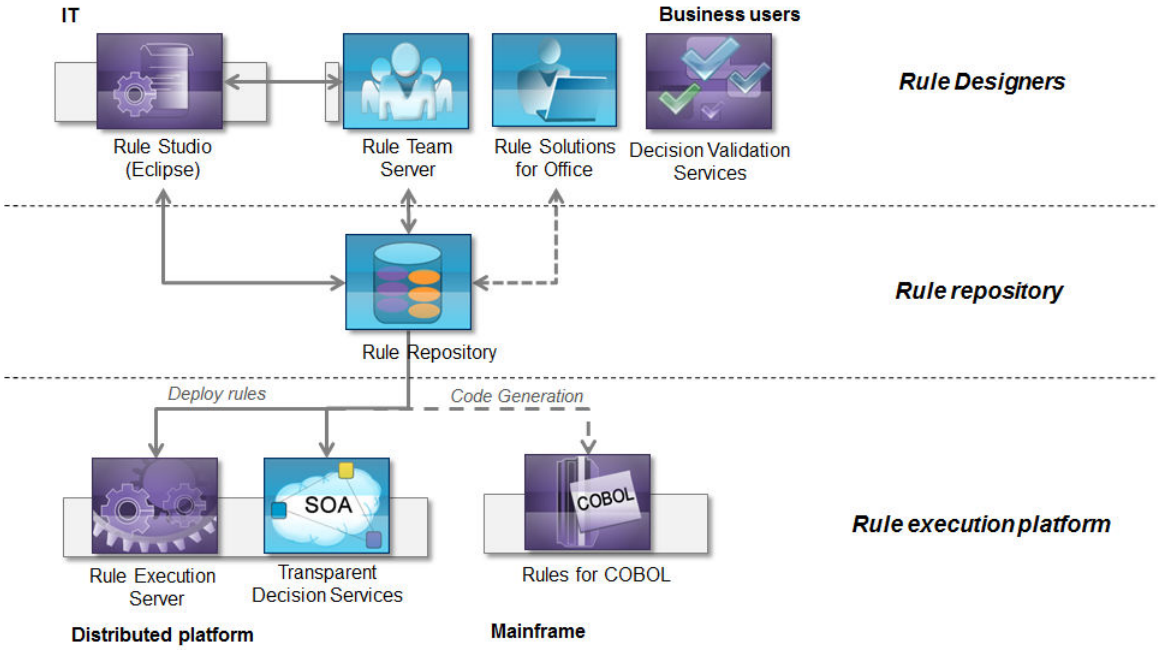


ILOG JRules od spoločnosti IBM patrí medzi rozšírené nástroje tohto druhu a na ich implementáciu využíva jazyk Java, ako aj prispôsobené prostredie IDE Eclipse. Prostredie JRules sa delí na 3 základné vrstvy:

- Návrhová vrstva - rule designers
- Centrálne úložisko pravidiel - rule repository
- Platforma pre nasadenie aplikácií - rule execution platform

Pre lepší prehľad aj pochopenie viď obrázok 2.3.

Obr. 2.3: Oddelenie vývoja pravidiel od celkového vývoja aplikácie



Podrobnejší popis funkcionality ILOG Jrules ako aj zmyslu jednotlivých platforiem nie je účelom mojej diplomovej práce a nebudem ho preto uvádzať.

2.3 IBM WebSphere Message Broker

IBM WebSphere Message Broker je komplexný nástroj slúžiaci na sprostredkovanie informácií. Zdroj [3] ho definuje takto: „WebSphere Message Broker je silný informačný sprostredkovateľ, ktorý umožňuje obchodným dátam vo forme správ tiecť medzi oddelenými aplikáciami a medzi mnohými hardvérovými a softvérovými platformami. Na jednotlivé dáta, ktoré prechádzajú cez Message Broker, sa dajú uplatniť rôzne pravidlá, aby ich bolo možné nasmerovať, ukladať, získavať a meniť."
V tejto časti práce popíšem WebSphere Message Broker všeobecne, od možností jeho nasadenia až po konkrétnu funkcionalitu využitú v systéme včasného varovania, aj s opisom jej implementácie v samotnom nástroji. Po opisoch jednotlivých funkcií bude nasledovať aj porovnanie s alternatívami v rámci IBM produktov i v rámci celého trhu (napríklad spoločnosti Oracle).

2.3.1 Stručná charakteristika IBM WebSphere Message Broker

Nástroj IBM WebSphere Message Broker sa skladá z niekoľkých prostredí a IDE (integrated development environment), ktoré sú založené na Eclipse. Spoločne slúžia na komplexnú správu celej platformy.
WebSphere Message Queue Explorer je nástroj pre vytváranie, úpravu a kontrolu brokerov - sprostredkovateľov, na ktorých sa nasadzujú jednotlivé toky správ. Broker beží autonómne, má k sebe priradené porty, loguje správy, ktoré k nemu prichádzajú, vrátane všetkých zmien v jeho nastaveniach, ktoré boli vykonané. Dá sa mu priradiť Message Queue Manager - správca fronty správ. Broker tak tvoria tieto fronty správ, skupiny vykonávania (execution groups) a tiež nastaviteľné servisy - configurable services. Častokrát broker komunikuje s databázou - podpora databáz je obšírnejšie popísaná nižšie.
WebSphere Message Broker Toolkit je taktiež nástroj založený na Eclipse, ktorý však prináša rozšírenú perspektívu pre vývoj Message Broker aplikácií. Väčšina nižšie popísaných operácií sa vykonáva práve v ňom. Toolkit slúži najmä na budovanie tokov správ, ktoré sú potom nasadzované na brokerov. Okrem toho má špeciálne nástroje na test tokov správ, budo-

vanie servisov, tvorenie XML, XSD schém a iných.
Na udržiavanie verzií a inštalácií systémov slúži Installation manager. Možná je aj manuálna kontrola a modifikácia parametrov cez konzolu, či test fronty správ cez pomocné programy.

2.3.2 Využitie a integrácia služieb

Message Broker ponúka nasledovné funkcie:

- **Univerzálna prepojitelnosť** - zjednodušená prepojitelnosť medzi jednotlivými aplikáciami umožňuje efektívne vytváranie flexibilných a dynamických infraštruktúr.
- **Smerovanie a zmena rôznych typov správ**:
  - podpora širokej škály protokolov, medzi ktoré patrí aj WebSphere MQ (Message Queue), $Java^{TM}$, Message Service (JMS), HTTP(S), Web servisy (SOAP a REST), Súbory, TCP/IP, Podnikové informačné systémy (vrátane Siebel a SAP), Servisnú komponentnú architektúru (Service Component Architecture - SCA) a iné.
  - podpora väčšiny rozšírených dátových formátov, napríklad binárny (C/COBOL), XML, IDoc, CSV (Comma separated values - čiarkou oddelené hodnoty) a iné.
  - podpora operácií a interakcií s dátami v podobe správ, ktoré umožňujú ich smerovanie, filtrovanie, zmenu, obohatenie, monitorovanie, distribúciu, dekompozíciu, koreláciu, odhaľovanie, usporadúvanie a iné.
- **Jednoduché programovanie** - pri transformácií správ je možné použiť grafické mapovanie, jazyk Java, rozšírený jazyk SQL (ESQL), Post Hypertext Preprocessor (PHP), Extensible Stylesheet language (XSL) a WebSphere Transformation Extender (WTX). Systém umožňuje takisto znovu-použitie určitých komponentov, napríklad prácu s WebSphere MQ, konvertovanie týchto správ do súborov, rozbaľovanie obálok (envelope) pri SOAP správach a podobne.
- **Operačný manažment a výkon** - pre rôzne operačné systémy a hardvérové platformy poskytuje Message Broker rozsiahle administratívne a na správu systému určené nástroje.

- **Podpora pre adaptéry a súbory** - adaptéry sú v rámci nástroja Message Broker dostupné v podobe zabudovaných tokov správ, ktoré poskytujú pripojenie k trom hlavným podnikovým informačným systémom - SAP, Siebel a PeopleSoft. Okrem toho umožňuje správu a spracovanie súborov, vrátane protokolov FTP a SFTP (protokol pre prenos súborov a zabezpečený protokol pre prenos súborov), cez zabudované prvky pre toky správ.

2.3.3 Práca s databázou

Definícia prístupu k databáze

Medzi dôležité súčasti integračných nástrojov patria takisto rozhrania na prácu s databázami. Message Broker umožňuje napojenie na väčšinu využívaných databáz, akými sú napríklad Informix, SQLite, MySQL, Sybase, PostgreSQL, Oracle či DB2.

Databázové definície Na získanie informácií o databáze k určitému časovému okamihu sa v tomto nástroji využíva databázová definícia (database definition file), súbor, ktorý využíva samostatne definované pripojenie. Pripojenie definuje, na ktorú konkrétnu databázu (podľa názvu a adresy) sa pripája a pod akým užívateľom a heslom, prípadne inou formou prihlasovania (napríklad LDAP).
Okrem pripojenia sa ďalej určuje, ku ktorej schéme v databáze sa bude pristupovať a aké práva budú v rámci tejto definície priradené. Obmedzia sa tak potenciálne zbytočné zásahy do štruktúry tabuliek, ak plánujeme len pristupovať k záznamom a podobne.
Takto vytvorená databázová definícia bude potom obsahovať štruktúru jednotlivých tabuliek a pohľadov v rámci schémy. Ak sa pridá medzi referencie projektu, je možné cez ňu definovať databázové dotazy. Nevýhodou je, že nezachytáva zmeny v databáze, ktoré nastanú od jej vytvorenia a tak sa musí manuálne aktualizovať.

Povolenie prístupov k databáze Samotná definícia databázy však nie je všetkým, čo treba vykonať pre úspešné pripojenie sa k databáze v rámci toku správ v prostredí Message Broker. Ako už bolo vyššie spomínané, jednotlivé toky správ sa nasadzujú na tzv. broker, ktorý beží na

serveri a dá sa spravovať cez Message Broker Explorer. Jednotlivé brokery majú okrem svojich vlastných frontov správ či nasadených tokov takisto nastaviteľné servisy (configurable services). Servisov je k dispozícií veľké množstvo druhov a štandardne nie sú aktívne. Pre ich aktiváciu je nutné nastaviť niekoľko parametrov a manuálne ich vytvoriť.
Na pripojenie k databáze je nutný JDBC connector. Okrem toho sú medzi nastaviteľnými servismi aj napríklad pripojenia k SAPu, JMS, CORBA či CICS.
Konkrétne JDBC poskytovateľ obsahuje parametre potrebné pre vytvorenie tzv. URL pripojenia. Nastavuje sa tam názov databázy, schéma, typ, server, port a takisto bezpečnostná identita. Niektoré parametre vyžadujú zhodu v mene s inými nastaveniami v rámci toku správ. Jedná sa napríklad o pripojenia z Javy či v rámci grafických máp. Názov nastaviteľného servisu by sa mal pritom zhodovať s volaným JDBC pripojením či názvom databázovej definície.
V prípade nesprávneho nastaviteľného servisu alebo jeho absencie nie je možné korektne pracovať s obsahom databázy v rámci tokov správ.

Manipulácia s databázovým obsahom

Ak je definované spojenie - dokážeme sa pripojiť na lokálnu, respektíve vzdialenú databázu - máme práva, nastaviteľný servis a všetko ostatné je v poriadku, môžeme v rámci tokov správ pristupovať k obsahu databáz a prípadne ho modifikovať. V tejto časti prejdem cez jednotlivé alternatívy, ktoré na prácu s databázovým obsahom ponúka Message Broker.

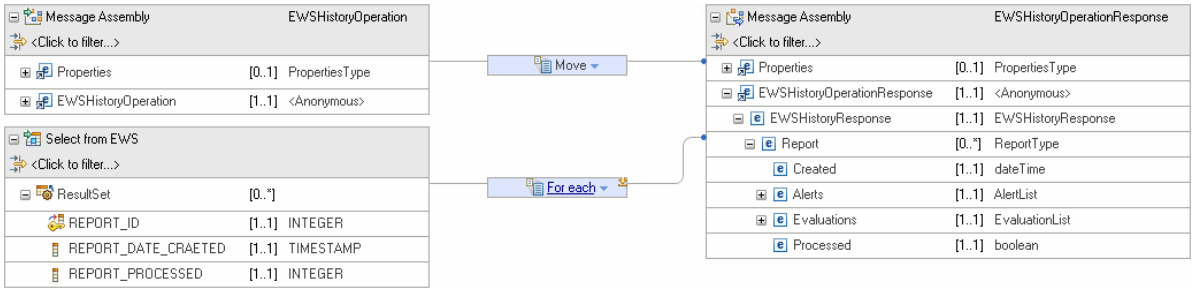
Grafické mapy Grafická mapa je preferovaným riešením transformácie dát na požadované typy a takisto na manipuláciu s ich obsahom. V posledných verziách nástroja Message Broker prechadázajú tieto mapy neustálym vývojom.
Mapa definuje jeden vstupný a niekoľko (najmenej však jeden) výstupných formátov správ. Poskytuje grafický prehľad tohto formátu, napríklad na základe XSD schémy, či definície Web servisu. Najpodstatnejšou časťou máp je však transformácia vstupného formátu na výstupný pomocou celej rady rôznych nástrojov, či už na prácu s dátovými typmi, cyklami alebo ďalšími pod-mapami.
V rámci mapovania je možné vykonať aj databázové operácie, ak má projekt prístup k databázovej definícií. Pri zmene formátov správ či databázovej definície sa tieto zmeny prejavia aj v

samotnej mape, bez nutnosti ručnej aktualizácie. Nástroj podporuje aj automatické mapovanie elementov na základe zvolených nastavení.

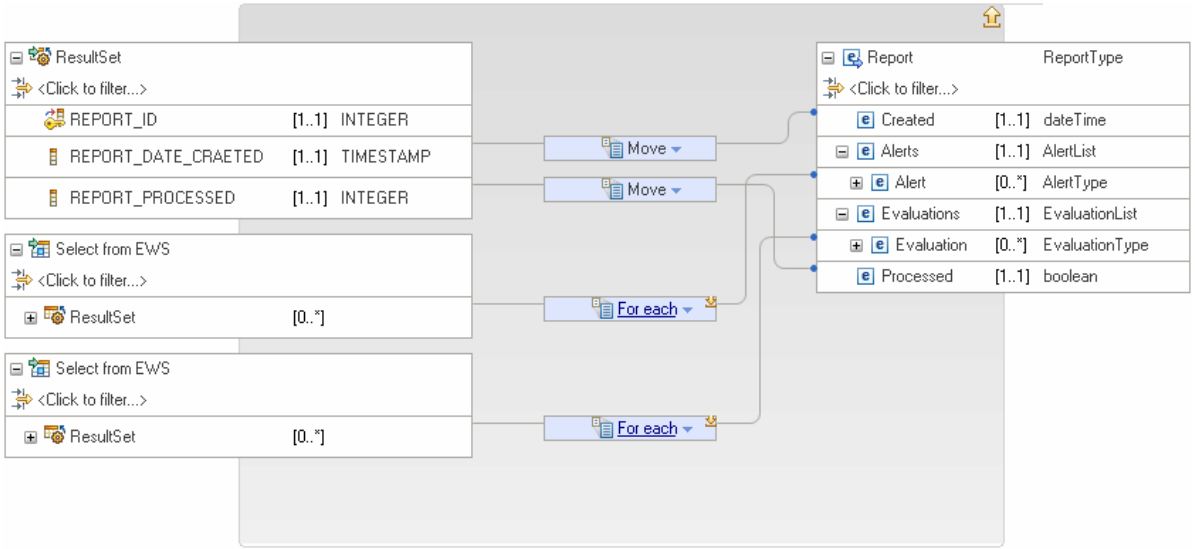
Mapa rozoznáva základné dátové typy a vie nad nimi robiť špecifické operácie. Takisto je možné mapovať N elementov do 1 prostredníctvom agregačných funkcií. Nad databázou sa pri mapovaní môže vykonať aj INSERT, DELETE alebo UPDATE jednotlivých riadkov.

Grafické mapy sú súčasťou nástroja Message Broker, ktorá je v mojej diplomovej práci využívaná vo veľkej miere, preto uvádzam aj obrázky z práce s nimi, ktoré demonštrujú funkcionalitu a prehľadnosť jednotlivých máp. Čo sa týka ich stability, sú ešte vo vývoji a nie všetky operácie sú dotiahnuté a fungujú správne.

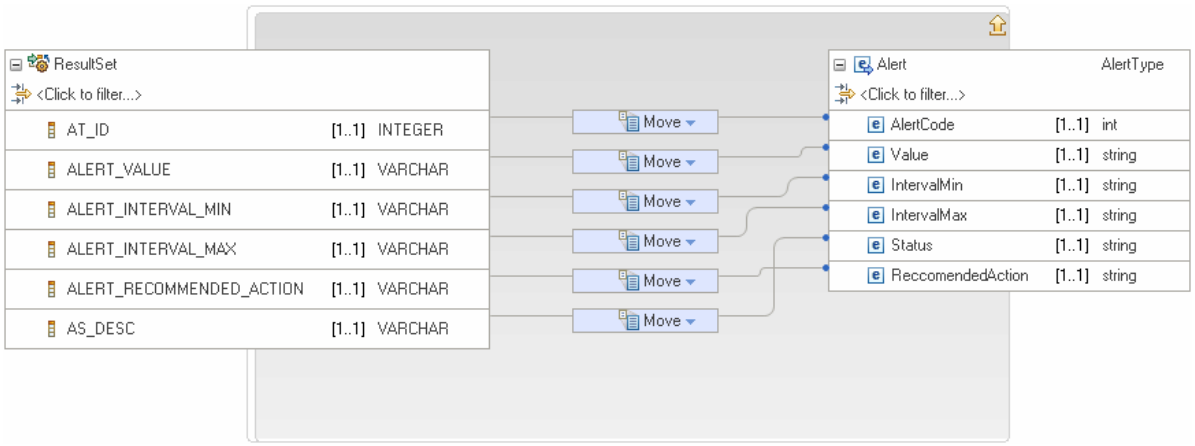
Obr. 2.4: Grafické mapovanie vstupného formátu na výstupný s použitím SELECTu



Obr. 2.5: Podmapa v cykle foreach



Obr. 2.6: Transformácia výsledkov SELECTu do elementov výslednej správy

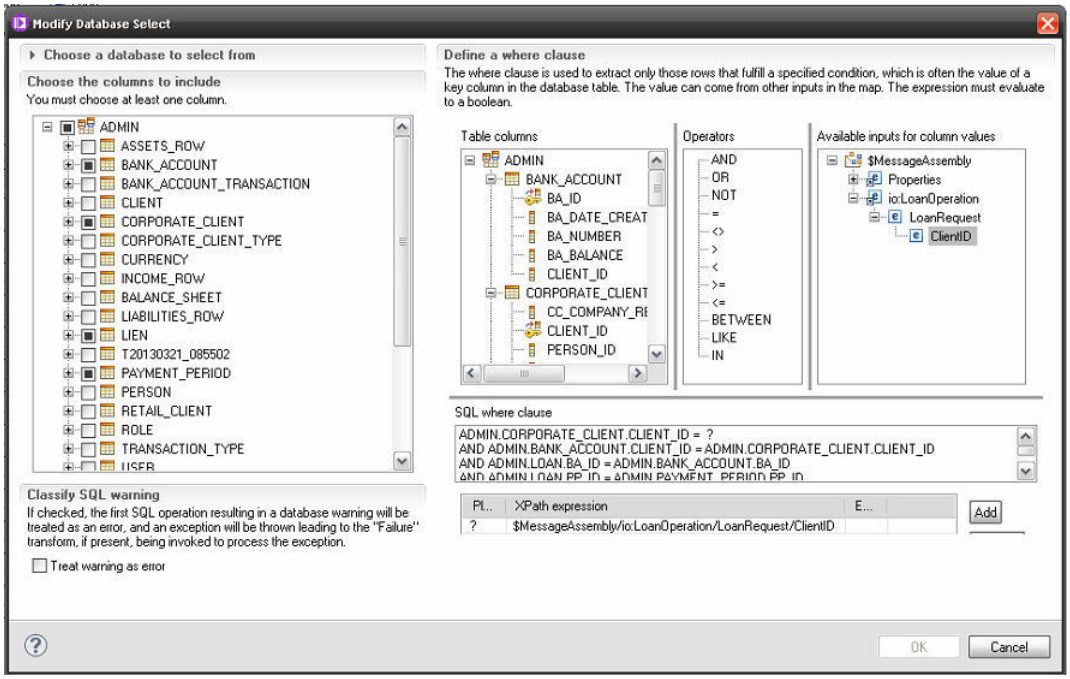


Prístup priamo z Javy Message Broker poskytuje dostatok možností na písanie vlastného kódu priamo v Jave. V rámci mapovaní je možnosť prevolávať vlastné statické metódy z tried, ktoré sú medzi referenciami projektu. Okrem toho existujú priamo triedy, ktoré slúžia na prácu so správami v rámci ich toku. Správy sa v nich dajú modifikovať, iba čítať a filtrovať alebo vytvárať.

Pripojenie k databáze sa dá realizovať cez JDBC pripojenie, ak sú splnené požiadavky, popísane v sekcií 2.3.3. Takýto prístup k databáze sa dá považovať za štandardný a nie je špecifikom nástroja Message Broker, preto ho nebudem hlbšie rozoberať.

SQL príkazy Aj napriek mnohým nástrojom, ktoré zaobaľujú prácu s databázou, je vždy dôležité umožňovať priame dotazy, akými sú SQL skripty. Message Broker prináša podporu na budovanie takýchto dotazov, kde sa dá priamo zvoliť databáza, schéma a tabuľka a vytvoriť SQL kód (pre ktorúkoľvek operáciu). Je prítomná podpora pre SQL či XQuery skripty. Samotná podoba skriptov a zásady pre ich písanie sú už štandardom a nie je nutné ich viac v mojej práci rozoberať.

Obr. 2.7: Operácia SELECT v grafickej mape



2.3.4 Vytvorenie servisov

V súvislosti so servisne orientovanou architektúrou a návrhom výsledného riešenia je dôležitá najmä funkcionalita na vytvorenie webových servisov, ktoré som popisoval vyššie, v kapitole 1.2.

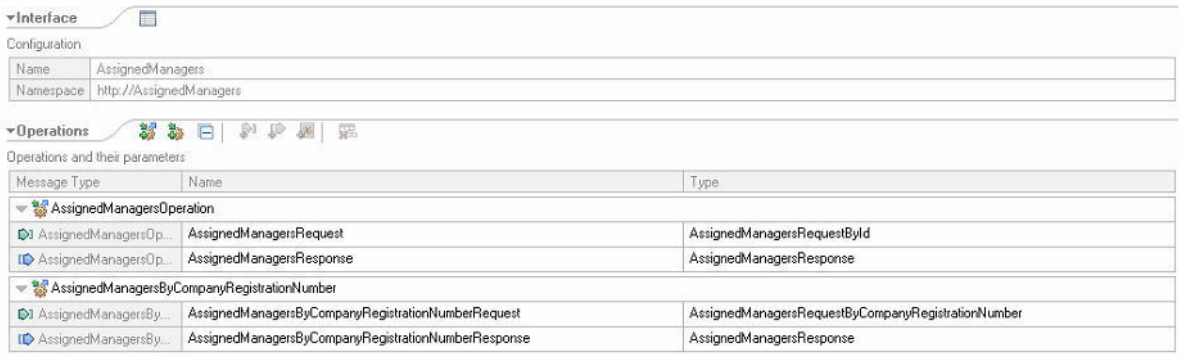
Message Broker ako integračná platforma má pre servisy podporu, dajú sa vytvárať pomocou špecifických projektov, ktoré riešia mnohé problematické časti ich implementácie za programátora a umožňujú tak, aby sa vývojár mohol v plnej miere sústrediť na logiku samotného servisu.

V servise sa určuje jeho názov, prefix, menný priestor a adresa. Potom ho tvoria jednotlivé operácie, ktoré môže vykonávať. Operácie môžu byť jednocestné alebo typu požiadavka - odpoveď. Pre každú operáciu sa potom definuje typ správy, ktorú prijíma a odosiela (v prípade jednocestnej operácie iba prijímanú správu).

Typy správ môžu byť primitívne alebo vlastné, definované užívateľom. Schému, ktorá definuje vlastnú správu, nie je nutné pripravovať vopred, ale dá sa vytvoriť priamo pomocou grafického editora. Jej súčasťou môžu byť ďalšie vnorené primitívne či vlastné typy elementov, dá sa definovať ich početnosť a iné vlastnosti.

Servisy sú nosnou časťou integrácie v systéme včasného varovania. Poskytujú v ňom jednotný a celistvý prístup k dátam, ktoré potom využíva obchodná vrstva - IBM BPM. Aj preto a pre lepšie pochopenie uvádzam grafický materiál, ktorý demonštruje ich funkcionalitu.

Obr. 2.8: Úprava servisu a jeho operácií



2.3.5 Porovnanie s inými možnými riešeniami na IBM platforme

Message Broker nie je jediným možným riešením pre integráciu medzi produktami od spoločnosti IBM. V rámci rodiny produktov WebSphere je ponúknutých hneď niekoľko alternatív, ktoré majú podobné vlastnosti, ale vždy aj určité špecifiká.

Jednou z týchto alternatív je AIS (advanced integration service), implementovateľný vo WebSphere Integration developeri alebo JMS (Java message system), ktorý je API pre Javu.

Hlavným dôvodom, prečo bol Message Broker uprednostnený pred týmito systémami bola jeho podpora pre Web servisy. Servisy majú mnoho vlastností, ktoré vyhovujú architektúre nášho systému.

Samotné AIS nemá alternatívu ku všetkým funkciám, ktoré poskytuje Message Broker. Jeho implementácia je softvérovo náročná, keďže samotný Integration developer je súčasťou iného balíka ako Message Broker.

JMS je Java Message Oriented Middleware (MOM), pristupuje k správam na nižšej úrovni a preto nerieši za vývojára mnohé problémy, ktoré pri posielaní a prijímaní správ vznikajú. Ďalším argumentom proti použitiu JMS je aj menšia podpora tejto technológie v produktoch od IBM.

Inou možnosťou by bolo pristupovať k dátam priamo v IBM BPM a tak nevyužívať servisy, ktoré by poskytoval iný nástroj. Výhodou by bola menšia zložitosť riešenia, v skutočnosti by sme ani neintegrovali a dáta získavali v takej podobe, v akej ich priamo v tom momente potrebujeme.

Nevýhody takéhoto riešenia sú zjavné. Ak by sa menila databáza, museli by sme meniť každé jedno prístupové miesto a spôsob získavania dát. V komplexnej obchodnej vrstve by mohlo byť hľadanie takýchto miest veľmi náročné. Pri využívaní servisov by sa menilo len mapovanie v servise, obnovila sa databázová definícia a vyriešili sa problémy po jej zmene. Samotné volanie servisu by táto zmena neovplyvnila.

Argumentov za a proti jednotlivým nástrojom je určite viac a vyčerpávajúca analýza by bola časovo i technologicky náročná. Na základe dostupných informácií a možností sme z dostupných produktov zvolili práve Message Broker. Ďalšie súvislosti a výhody tejto voľby budú vysvetlene neskôr, pri opisovaní samotného riešenia.

2.3.6 Porovnanie s konkurenciou

Ak nie je Message Broker jediným možným riešením medzi produktami od IBM, dá sa očakávať, že aj konkurenčné spoločnosti vyvinuli podobné systémy, ktoré tiež riešia integračné problémy. Medzi najväčšie a najznámejšie patrí Oracle Service Bus.
Vyčerpávajúce porovnanie týchto systémov poskytuje zdroj [5]. Z tohto porovnania vyplýva, že Message Broker má „navrch“ v týchto oblastiach:

- Možnosti pripojenia
- Databázové operácie
- Možnosti integrácie (WebSphere MQ, SOA registre počas behu, obchodné udalosti)
- Ľahšie využitie adaptérov
- Výkon

Ďalšími alternatívami sú open source ESB systémy, ako napríklad Mule ESB a JBoss ESB. Podľa výskumu [6] je však ich výkonnosť značne nižšia ako pri použití platformy Message Broker. Oba porovnávané ESB systémy sú voľne dostupné, avšak ich výkonnosť je na úplne inej úrovni, vo výskume im bola dokonca daná výhoda niekoľkých JVM inštancií.
Message Broker dokázal spracovať v priemere 7359 správ za sekundu, zatiaľ čo Mule ESB 4669 a JBoss 524. Z výskumu tak vyplýva, že Mule ESB by vyžadovalo dvojnásobné množstvo hardvéru, aby konkurovalo záťažou platforme Message Broker. JBoss by hardvéru potreboval dokonca 15-násobné toľko. Konkrétne konfigurácie a verzie jednotlivých nástrojov sú uvedené v spomenutom dokumente.
Na záver je nutné dodať, že výskumy a ich výsledky sú pre nás iba orientačné, keďže všetky spomenuté systémy sú v neustálom vývoji a tak by podobné testy v súčasnosti mohli dopadnúť úplne inak. Avšak pri počiatočnom rozhodovaní boli tieto fakty dostatočné pre racionalizáciu rozhodnutia sa pre nástroj Message Broker.

Kapitola 3

Analýza problému

Problémom, ktorým sa moja diplomová práca zaoberá je default klienta a jeho následky pre banku. Vo všeobecnosti má banka 4 druhy klientov:

- **Retail klienti** - fyzické osoby, prinášajú menší obnos peňazí, sú im poskytované obyčajné služby.
- **Korporátni klienti** - firmy, právnické osoby, prinášajú väčší obnos peňazí, sú im poskytované investičné a obchodné úvery.
- **Inštitúcie** - iné banky, regióny, verejný sektor.
- **Suveréni** - štáty a medzinárodné banky.

Systém včasného varovania sa bude zameriavať najmä na korporátnych klientov. Retail klienti sú v tomto ohľade menej zaujímaví, ich úvery rádovo nižšie a banka má menej možností jednať. Pre väčšie celky, akými sú inštitúcie či štáty je automatizácia procesov takmer nemožná a rozhodovanie je i vzhľadom na malý počet takýchto klientov (a značne vyššie finančné čiastky) riadené inými pravidlami.
Klientov bez úveru nebudeme v nami vyvíjanom systéme monitorovať, pretože varovanie o ich finančnej situácií nemá pre banku praktický význam.
V tejto kapitole postupne rozoberiem príčiny defaultu klientov, aké sú možnosti na jeho predpovedanie a ako sa dá toto riziko defaultu menežovať.

3.1 Ako funguje banka

Ako už bolo spomenuté, moja diplomová práca sa zaoberá systémom monitorovania portfólia klientov banky. Samotná banka je finančná inštitúcia, ktorá prijíma vklady od zákazníkov, poskytuje úvery a ďalšie služby (ako napríklad finančné poradenstvo), nakupuje rôzne finančné aktíva a iné.
Vklady klientov sú splatné na požiadanie alebo v krátkej výpovednej lehote. Hlavným účelom banky ako právnickej osoby je generovať zisk.
Zdrojom zisku pre banku sú:

- úroky,
- poradenstvo,
- vedenie účtu a iné poplatky,
- vlastné investície.

Poplatky sú pre banku zdrojom, ktorý nie je priamo vystavený riziku a dá sa jednoduchšie určiť jeho veľkosť. Pri všetkých ostatných zdrojoch je nutné operovať s rizikom, ktoré výrazne ovplyvňuje konečné generovanie zisku.
Pojem riziko je veľmi široký a v banke odlišujeme niekoľko základných typov. Riziko znamená inými slovami aj mieru neistoty, šancu, že predpokladaný výsledok nenastane. V prostredí bánk rozoznávame riziká:

- **Kreditné riziko** - jedná sa o zlyhanie dôvery, druhá strana nedokáže splácať dlžnú čiastku podľa dohody, kľúčové riziko pre banky.
- **Riziko trhu** - trhom vnímaná cena aktív je odlišná od nami predpokladanej ceny, vzniká pri investíciach na finančnom trhu.
- **Riziko likvidity a fundingu** - strata schopnosti banky dodržať svoje finančné záväzky, v závislosti od času odlišujeme riziko likvidity a fundingu.
- **Operačné riziko** - zlyhanie systémov, výpadky hardvéru, ako napríklad výpadok elektriny, požiar, zemetrasenie a iné.

Risk manažment je snaha o pochopenie a identifikáciu rizika, nie o jeho minimalizáciu či likvidáciu. Pochopenie a identifikácia rizika v konečnom dôsledku generuje banke najväčší zisk.

3.2 Default klienta a poskytovanie úverov

Z predchádzajúcej kapitoly je evidentné, že práca s rizikom je pre banku nesmierne dôležitá. Chcel by som však objasniť, ako je v podstate vôbec možné s ním pracovať a čo to default klienta vlastne je.

Default je stav, keď klient nedokáže dlhšie splácať svoje záväzky v dohodnutej podobe. Znamená to, že v banke, ktorá mu poskytla svoje zdroje, dôjde k strate, ak sa táto situácia klienta nezlepší. V tomto stave sa klient nachádza, ak je platobne neschopný (hodnota jeho záväzkov prevyšuje hodnotu jeho majetku a ak nie je schopný plniť aspoň jeden platobný záväzok, 30 dní po lehote splatnosti).

V takomto prípade sa situácia čiastočne rieši konkurzom, kedy sa speňaží jeho majetok a uspokojujú sa veritelia tohto dlžníka. Konkurz je vyhlásený súdom. Avšak, jedná sa len o pomerné uspokojenie veriteľov, majetok dlžníka v čase bankrotu určite nemôže vyrovnať všetky jeho dlhy a tak dochádza k strate.

Ak by banka poskytovala úvery bez úroku, prípadne by toto riziko do úverov nezapočítavala, nebolo by pre ňu výhodné poskytovať takéto služby. Avšak s rizikom defaultu je možné kalkulovať a pripočítavať ho v podobe úroku k jednotlivým splátkam. Tieto možnosti budem demonštrovať na jednoduchom príklade.

Banky už pri poskytovaní úveru zaradia klienta na základe informácií o ňom do istej rizikovej skupiny. Pre túto skupinu je zadefinovaná určitá hodnota *PD* (probability of default). Ak by táto skutočnosť nastala, odhaduje sa strata *LGD* (loss given by default). *EAD* (exposure of default) je množstvo, ktoré bolo poskytnuté klientovi v čase defaultu. *EL* (expected loss), strata, ktorú vždy pri poskytovaní úverov očakávame.

$PD = 2\%$

$LGD = 40\%$

$EAD = 100€$

$$EL = PD \cdot LGD \cdot EAD = 100€ \cdot 0.4 \cdot 0.02 = 1€ \tag{3.1}$$

Takto vieme určiť očakávanú stratu. Túto stratu je potrebné minimálne pokryť navýšením úrokovej miery. Úroková miera sa označuje ako *IR* a *s* je sadzba, o ktorú chceme úrokovú mieru navýšiť. Dá sa to určiť aj takýmto spôsobom:

$IR = 2\% + s$

$EAD = 1€$

$$(1 - PD) \cdot (EAD + s) = EAD \tag{3.2}$$

$$(1 - PD) \cdot (1 + s) = 1 \tag{3.3}$$

$$1 - PD = \frac{1}{1+s} \tag{3.4}$$

$$s = \frac{1}{1-PD} - 1 = \frac{PD}{1-PD} = \frac{2\%}{98\%} = 0.02041 \tag{3.5}$$

Podľa pravdepodobnosti defaultu sa tak portfólio klientov rozdeľuje na určité kategórie, pre ktoré sú rôzne úrokové miery. Zisk, ktorý pri poskytovaní úverov banka generuje je tak ovplyvnený do značnej miery úrokovou mierou. Vysoká úroková miera bude určite odrádzať klientov, nízka naopak nepokryje predpokladané straty.

3.3 Možnosti banky pri odhalení defaultu

Dôvody, pre ktoré je dôležité odhaliť default klienta som popísal v predchádzajúcej kapitole. Veľmi dôležité sú metódy, ktorými je možné tento jav predpovedať s určitou pravdepodobnosťou, na základe dát, ktoré máme o klientovi k dispozícií.

Informácie, ktoré budú pre systém včasného varovania kľúčové, sú:

- Transakčné dáta - pohyby na účtoch, všetky klientovi debety a kredity, vrátane splátok úveru.
- Finančné výkazy - v rámci systému včasného varovania sa budú overovať výkaz ziskov a strát a súvaha. Oba výkazy sú odovzdávané pravidelne a dá sa nad nimi vykonávať analýza. Nevýhodou je ich značné oneskorenie.
- EWS história klienta - história klienta v samotnom systéme, kedy a ako bol posudzovaný, kto ho posúdil a ako.
- Informácie o úvere - čím klient za daný úver ručí, ako a kedy ho spláca, aká časť úveru bola doteraz zaplatená a iné.
- Neurónové siete - poskytujú možnosť matematicky podloženej predikcie správania sa klienta, avšak vyžadujú väčšie množstvo dát (ak chceme, aby ich výsledky boli presné) a ťažko sa z nich vyvodzujú závery.

Na základe určitých pravidiel a modelov je možné z týchto dát vyvodiť určité závery o klientovej situácií. Tieto závery nazveme **výstrahami** alebo **alertami**. Podrobnejšie budú jednotlivé výstrahy rozobraté v ďalších kapitolách. Ich úlohou je jasne indikovať, čoho sa klient dopustil.

Ich udelenie môže byť automatické alebo manuálne. Automatické v prípade, že je možné na základe určitých faktov ihneď vyvodiť záver, napríklad v prípade, že nie je evidovaná splátka v riadnom termíne.

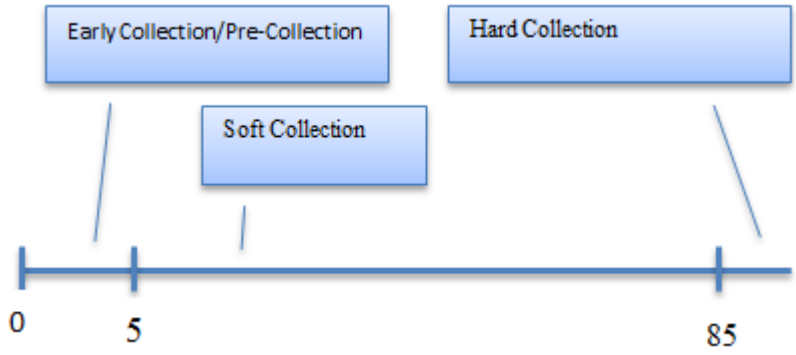
Samotné metódy na vyvodenie týchto záverov nie sú predmetom mojej záverečnej práce, sú však dôležitou súčasťou systému včasného varovania.

Ešte je dôležité položiť si otázku: „Na čo nám všetky tieto informácie vlastne sú? Aké má banka reálne možnosti ovplyvniť klienta, ak zistí, že sa jeho situácia zhoršila?“

Ak by neboli žiadne právne prostriedky, ktoré by umožnili banke konať a ovplyvniť klienta pred tým, ako sa rozhodne o konkurze, strácal by samotný systém svoj význam. Avšak v skutočnosti je týchto možností veľké množstvo a prax ukazuje, že je dôležité ich nepodceniť, komunikovať s klientom a snažiť sa mu pomôcť. Pomôcť sa mu dá aj v podobe poradenstva,

prenastavenia úveru, zmeny jednotlivých splátok a iných akcií.
V banke sa klient radí do určitých skupín, na základe toho, koľko dní mešká s platbou.

Obr. 3.1: Rozdelenie klienta na základe počtu dní omeškania so splátkou



Pre jednotlivé skupiny a k nim naviazané obdobia platí:

1. **Early collection / pre-collection** - nie je vyvíjaná žiadna aktivita, banka nemá možnosti. Toto obdobie sa nazýva aj akceptovateľná posplatnosť.
2. **Soft collection** - klient je pridaný do zoznamu monitorovaných klientov, stále nie je možnosť vyvíjať akcie.
3. **Hard collection** - klient je kontaktovaný bankou a situácia sa aktívne rieši.

Možnosti banky sú zmluvne zachytené pri podpisovaní úveru. Reálne možnosti banky sú takéto:

1. zníženie limitov prečerpávania úveru,
2. odobratie limitu ako takého,
3. pridať dodatočný kolaterál (dodatočné ručenie za úver),
4. reštrukturalizácia úveru,
5. odklad platieb,
6. odpis pohľadávky (zapísané ako strata).

Každý klient musí byť posudzovaný aj z hľadiska veľkosti úveru a možnej škody pre banku. Ak by náklady na hard resp. soft collection preverili možnú získanú čiastku, tak je vykonanie týchto akcií bezpredmetné.

Ako budú tieto analyzované javy zapracované a zaradené do systému včasného varovania banky uvediem v kapitole 4.

Kapitola 4

Návrh riešenia - Systém včasného varovania banky

V tejto kapitole opíšem navrhovaný systém včasného varovania (early warning system - ďalej len EWS). V kapitole 3 som vysvetlil pojem default klienta, zdroje zisku banky a jeho hlavné obmedzenia. Z uvedených skutočností vyplýva potreba kvalifikácie risku pre jednotlivých klientov, nutnosť ich pravidelnej kontroly a poskytovania čo najväčšieho počtu informácií, ktoré by umožnili zodpovedným osobám vykonať rozhodnutia v prípade zmeny situácie klienta.
EWS na základe rôznych príznakov v dohľadnom čase identifikuje zmeny v okolí ako aj v samotnom vnútri klienta, ktorý je zákazníkom banky. Jedná sa o zmeny, ktoré ovplyvňujú podnik a jeho schopnosť ďalej fungovať v rámci svojich medzí (a v konečnom dôsledku aj splácať banke úver).
Pod pojmom EWS sa rozumie informačný systém - aplikácia - ktorý je schopný vopred upozorniť na blížiacu sa zmenu. V trhovom prostredí, kde dochádza k častým zmenám, je kľúčové získať informácie čo najskôr a dokázať sa rýchlo prispôsobiť zmenám.
Úlohy EWS by sa dali zhrnúť do týchto bodov:

- „Včasné“ zaregistrovanie zmien klienta, v jeho okolí i vo vnútri.
- Automatický a rýchly prenos informácií kompetentným osobám, v našom prípade zamestnancom banky.

- Identifikácia, analýza a vyhodnotenie zmien a ich rizika.
- Poskytnúť informácie, prípadne automaticky rozhodnúť o závažnosti zmien a ich vplyvu na budúcnosť.

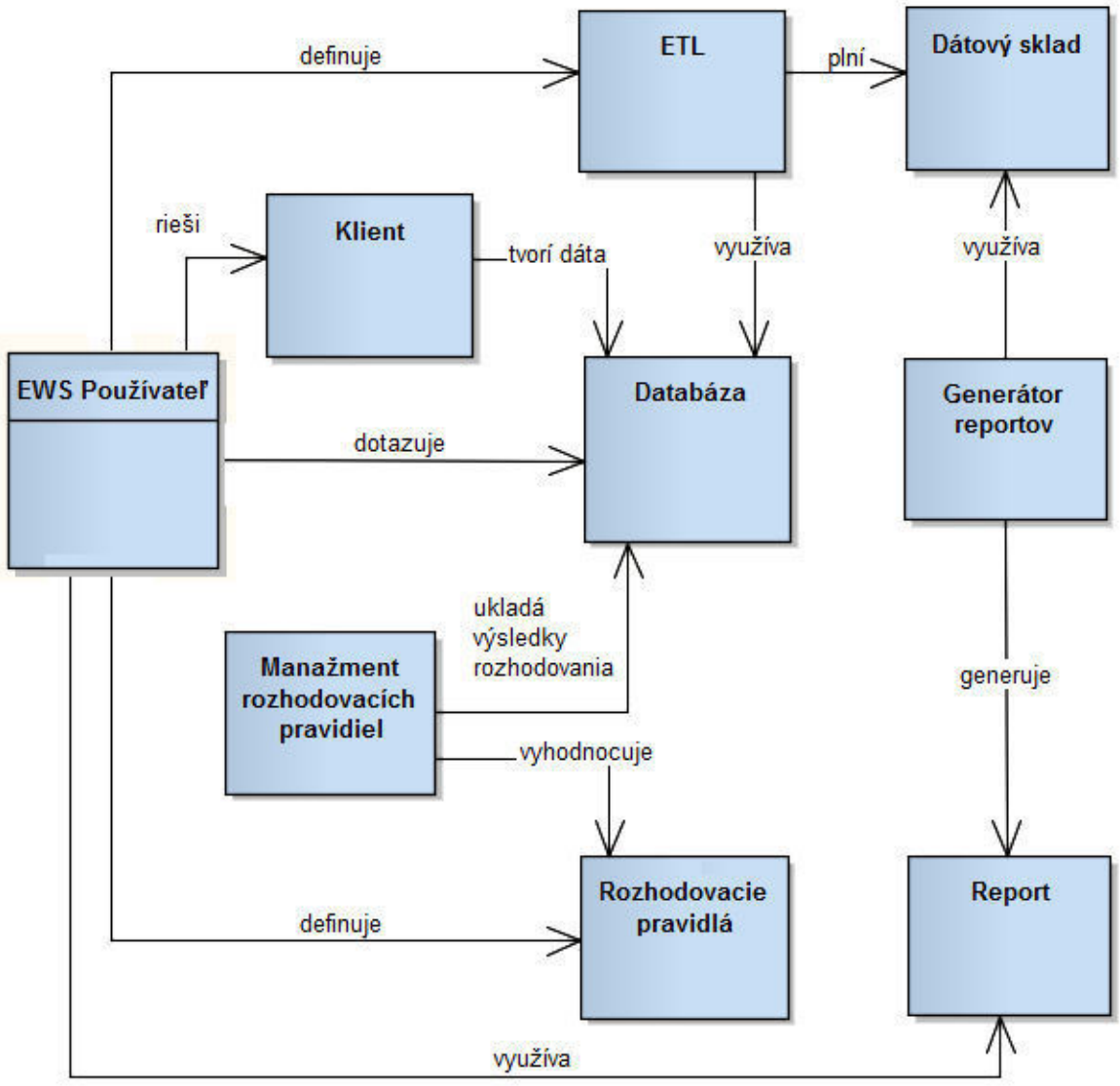
Konkrétnu implementáciu tejto funkcionality rozoberiem ďalej v tejto kapitole.

4.1 Konceptuálny model riešenia

Konceptuálny (alebo doménový) model je podľa publikácie [10] spôsobom na zachytenie myšlienok a nápadov v problémovej oblasti. Konceptuálny model je na rozdiel od ostatných druhov modelov (napríklad logického či fyzického modelu) implicitne nezávislý od dizajnových či implementačných záležitostí.
Cieľom je vyjadriť význam jednotlivých pojmov a konceptov a nájsť medzi nimi správne vzťahy.
V nami navrhovanom systéme sa nachádzajú EWS používatelia, ktorí v závislosti od svojej role vykonávajú dotazy na databázu, definujú ETL proces, ktorý slúži na načítanie, zmenu a spracovanie dát pre dátový sklad.
EWS používateľ má k dispozícií sadu reportov, ktoré sú výstupom systému a na základe informácií, ktoré poskytujú, môže postihnúť zákazníka banky, prípadne vykonať inú akciu, v závislosti od situácie.
Samotný zákazník je v systéme zachytený len z hľadiska jeho interakcií s databázou, pre ktorú svojou činnosťou „tvorí“ dáta.
V jadre systému sa nachádza databáza, úložisko všetkých podstatných informácií o klientoch a ich činnosti. Okrem toho sa v databáze ukladajú výsledky pravidiel. Pravidlá sú jasne odlíšené od databázy a budú vykonávané osobitným systémom. Ich výsledky sú kľúčové a prechádzajú na entitu manažment rozhodovacích pravidiel. Pravidlá sú taktiež definované používateľmi EWS.
ETL proces čerpá dáta najmä z databázy, ako aj z iných zdrojov, ktoré kvalifikujeme ako externé a nie sú v tejto fáze systému dôležité. Dátový sklad bude špeciálne navrhnutým skladom pre EWS, nie už existujúcim skladom, využívaným v banke.

Dátový sklad bude slúžiť na generovanie reportov. Generátor reportov bude odosielať komplexné dotazy na sklad a generovať definované reporty vzhľadom na stav klienta a zmenu, ktorá nastala. Report bude ďalej obsahovať súbor alertov, na základe ktorých bude možné vyhodnotiť situáciu.

Obr. 4.1: Konceptuálny model riešenia



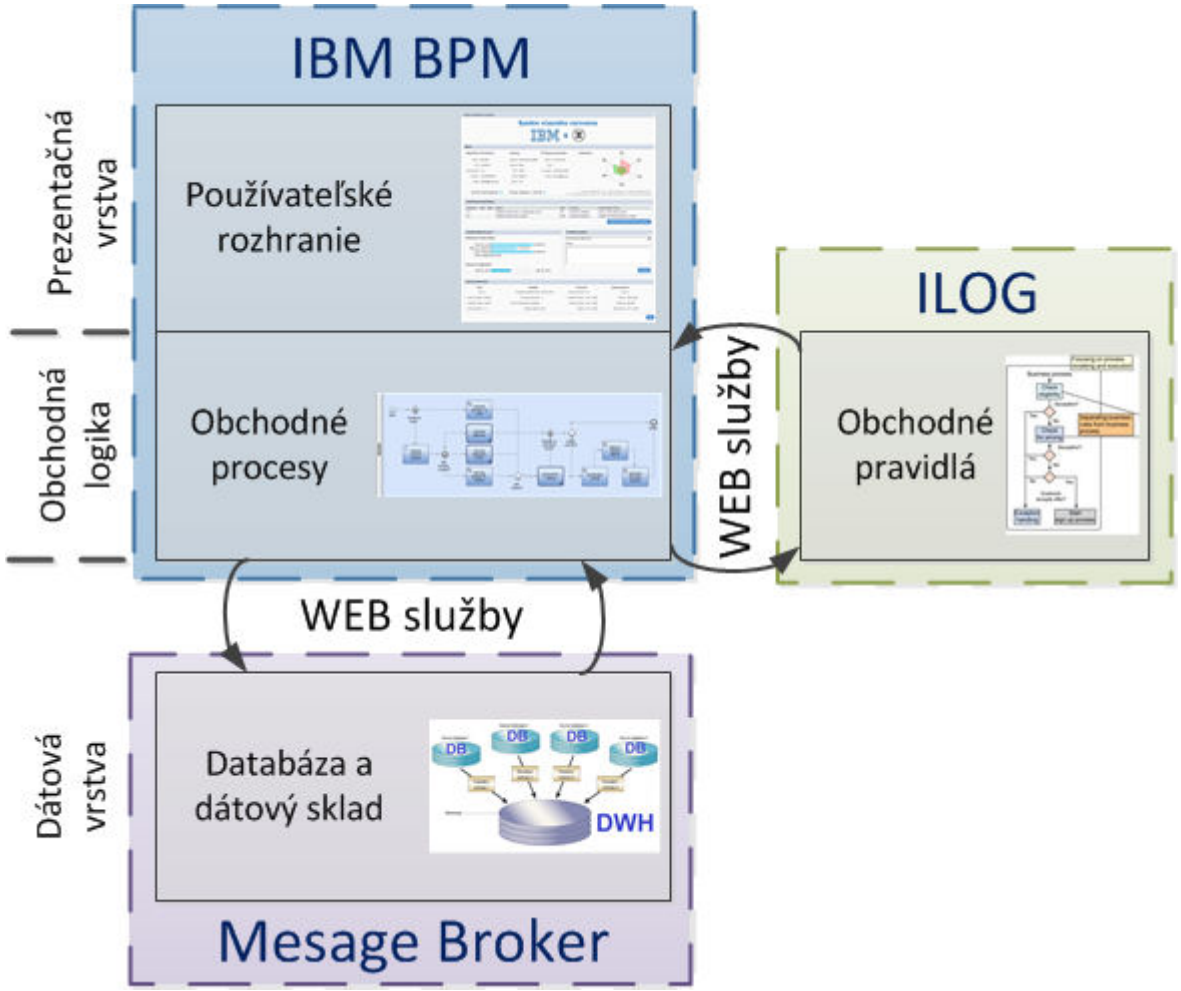
V kapitole 2 boli spomenuté 3 nástroje z rodiny produktov WebSphere. Každý z nich má svoje zastúpenie aj v uvedenom konceptuálnom modeli a ich správne využitie je kľúčové pre celý EWS.
Štúdium a analýza jednotlivých nástrojov boli podstatnou časťou mojej diplomovej práce.

- **IBM WebSphere ILOG** umožňuje manažment rozhodovacích pravidiel a vyhodnocuje samotné rozhodovacie pravidlá.
- **IBM WebSphere BPM** poskytuje správu jednotlivých reportov EWS používateľom a generuje ich s využitím servisov.
- **IBM WebSphere Message Broker** zastrešuje komunikáciu medzi jednotlivými súčasťami systému a pomocou servisov poskytuje dáta obchodnej vrstve (BPM).

Pre lepšie vysvetlenie jednotlivých väzieb a využitia nástrojov, ako aj na rozdelenie úloh v rámci nášho tímu pripájam obrázok 4.1.

Dátová vrstva je v našom systéme tvorená databázou a dátovým skladom. Viac informácií o ich konkrétnej podobe a využití uvádzam v kapitolách 4.2 a 4.4.
Dáta z databázy a dátového skladu budú prakticky poskytované obchodnej vrstve (vrstve logiky) pomocou servisov, ktoré budú tvorené v nástroji IBM WebSphere Message Broker a nasadené na spoločnom brokeri, v rozličných skupinách vykonávania.
Návrh a implementácia dátovej vrstvy ako i servisov nad nimi boli hlavným účelom mojej diplomovej práce, ako i práce môjho spolužiaka, Juraja Branického. Tvorili sme tiež spoločne konceptuálny model a kostru celkového riešenia, ktoré vznikalo v 5-člennom tíme.
Logickú vrstvu tvoria jednak obchodné procesy, implementované v nástroji IBM WebSphere BPM, ako aj rozhodovacie (obchodné) pravidlá, nasadzované, spravované a uložené v IBM WebSphere ILOG. Obchodné procesy získavajú všetky potrebné dáta z web servisov z dátovej vrstvy a poskytujú ich jednak na prehľad používateľovi, ako aj pre obchodné pravidlá do ILOG-u. Komunikácia medzi BPM a ILOG-om prebieha taktiež pomocou web servisov, tvorených v nástroji ILOG.
Prezentačná vrstva je tvorená najmä v nástroji BPM, kde sú definované obrazovky pre jednotlivé role pomocou web stránok. Reporty využívajú okrem HTML aj javascript a iné technológie previazané s HTML na strane klienta.

Obr. 4.2: EWS a využitie jednotlivých nástrojov



Prácu v nástroji BPM tvoria a píšu študenti Boris Bučko a Vladimír Hanušniak, obchodné pravidlá a servisy vo WebSphere ILOG implementuje posledný člen tímu, Igor Mäsiar.

EWS ako celok je však dielom rovnomernej práce každého z nás a skladal sa okrem návrhu, implementácie a testu demo verzie finálneho systému aj z procesu inštalácie a učenia sa práce v jednotlivých nástrojoch, rozpoznania ich možností a potenciálnych úloh v EWS a iných. Práca na dátovej vrstve so sebou niesla aj nutnosť tvorby testovacích dát.

4.2 Dátový model riešenia

Dátový model reprezentuje jednotlivé entity, ktoré vystupujú v systéme, vrátane ich vzťahov a atribútov.

V nami vytvorených modeloch sa nachádzajú tabuľky uchovávajúce často sa meniace transakčné dáta, ako aj pomerne stále číselníky. Číselníky sú odlíšené zelenou farbou a externé entity pre určité komponenty (tabuľky definované v inom komponente) sú vyvedené mimo hlavného diagramu.

Bankové účty sú jedným zo základných komponentov EWS. Bankový účet sa vždy viaže na klienta, ktorého v EWS evidujeme len v prípade, že má v danej banke účet. V iných prípadoch nemá zmysel udržiavať o ňom informácie.

Bankový účet sa aktualizuje po každej transakcií a určuje sa hodnota zostatku. Jedinečným identifikátorom je okrem interného ID aj číslo účtu.

Transakcie na danom účte zachytáva entita transakcie. Viaže sa na konkrétnu menu, ale vždy prenáša aj informáciu o pohybe v lokálnej mene - v našom prípade v Eurách. Transakciu ďalej určuje jej variabilný symbol, dátum a protiúčet. Informatívnou hodnotu je zostatok na účte po vykonanej transakcií.

Aby sa odlíšili rôzne typy transakcií, definuje sa číselník typ transakcie. V EWS sa jedná o kredit (navýšenie zostatku), debet (zníženie zostatku) a splátku (zníženie zostatku), ktorú je však nutné odlíšiť od obyčajných debetov. V EWS je esenciálne splátky monitorovať, sú základom aj pre mnohé výstrahy, rovnako ako sú pre ne základom aj transakcie vo všeobecnosti.

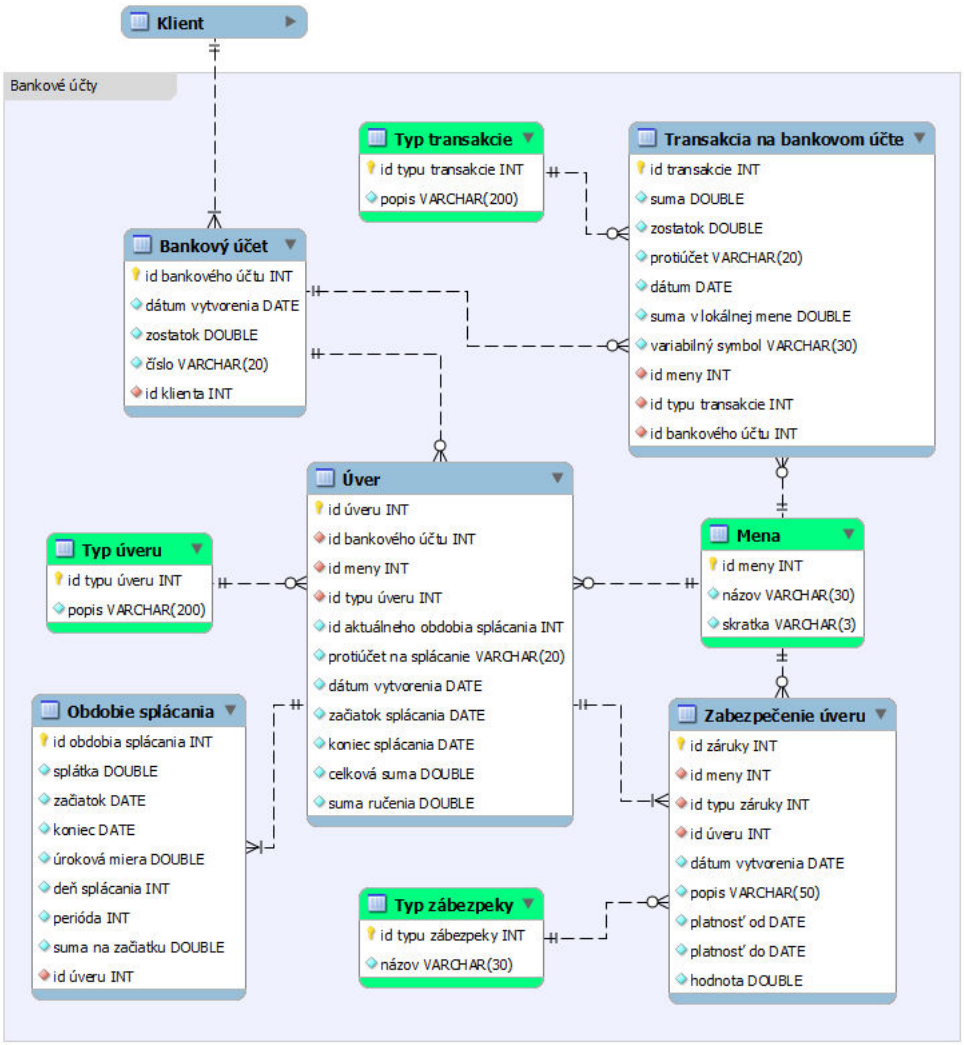
Dôležitou entitou je úver. V nami navrhovanom modeli sa úver vždy viaže na bankový účet v danej banke - je to bežná prax pri korporátnych klientoch, ktorá umožňuje banke sledovať väčšinu transakcií daného klienta.

Zábezpeka úveru je definovaná entitou zabezpečenie úveru, ktorá môže byť typu, ktorý definuje číselník typ zábezpeky. Zábezpeka bola odovzdaná v určitom čase a obsahuje podrobnejší popis, svoju platnosť a najmä hodnotu. Úver môže mať niekoľko zábezpek.

Samotný záznam v entite úver definuje protiúčet, na ktorý má byť splácaný, celkovú sumu, celkové ručenie, ale nehovorí nič o konkrétnych splátkach či úroku. Definuje však, ktoré obdobie splácania je práve aktuálne, odkedy sa začal splácať a kedy splácanie skončí.

Entita obdobia splácania určuje jednotlivé splátky úveru pre obdobia, ktorých môže mať úver niekoľko. Údaje v nej sú v rovnakej mene ako samotný úver. Obdobie je spravidla 3 roky, počas ktorého sa určí deň splácania (zvyčajne v polovici mesiaca) a perióda. Dá sa tak určiť, v ktorý deň už mala byť splátka uhradená. Taktiež je pre celé obdobie charakteristický určitý úrok a výška splátky.

Obr. 4.3: Dátový model - komponent bankové účty



Komponent klienti v sebe zahŕňa všetky potrebné informácie o evidovaných klientoch. Ako som už uviedol, klientom je v EWS ten, kto má v danej banke účet. Entita klient je predok entít korporátny a retail klient.

Obe entity majú spoločné informácie o adrese a je im v EWS priradený manažér. Manažér je zároveň používateľom EWS a má určitá rolu, viď komponenty nižšie. Eviduje sa pri ňom aj obdobie, kedy bol klientovi priradený, aby sa tak dala jednoznačne určiť zodpovednosť za konkrétne rozhodnutia.

Retail klient nie je pre EWS dôležitý a tvorí ho jedna konkrétna osoba. Osoba je však osobitná entita, pretože môže byť zároveň štatutárom korporátneho klienta (ale nemusí byť pri tom retail klientom banky).

Korporátny klient je hlavným bodom zájmu jednotlivých kontrol a predstavuje spravidla právnickú osobu. Dôležitým údajom sú okrem tých informatívnych IČO (identifikačné číslo organizácie) a DIČ (daňové identifikačné číslo).

Ďalej je korporát opísaný číselníkmi. Typ korporátneho klienta podáva dôležitú informáciu o právnej zodpovednosti, či sa jedná o akciovú spoločnosť, spoločnosť s ručením obmedzením alebo napríklad o samostatne zárobkovo činnú osobu.

Veľkosť je informatívna a podstatná pre rozhodovacie pravidlá, odlišujeme malú, strednú a veľkú spoločnosť, na základe metrík ako počet zamestnancov či veľkosť majetku.

Posledným číselníkom je SK NACE (predmet prevažujúcej činnosti), ktorý určuje oblasť, v ktorej korporát pôsobí. Dáta pre číselník boli spracované podľa súčasných dát zo štatistického úradu.

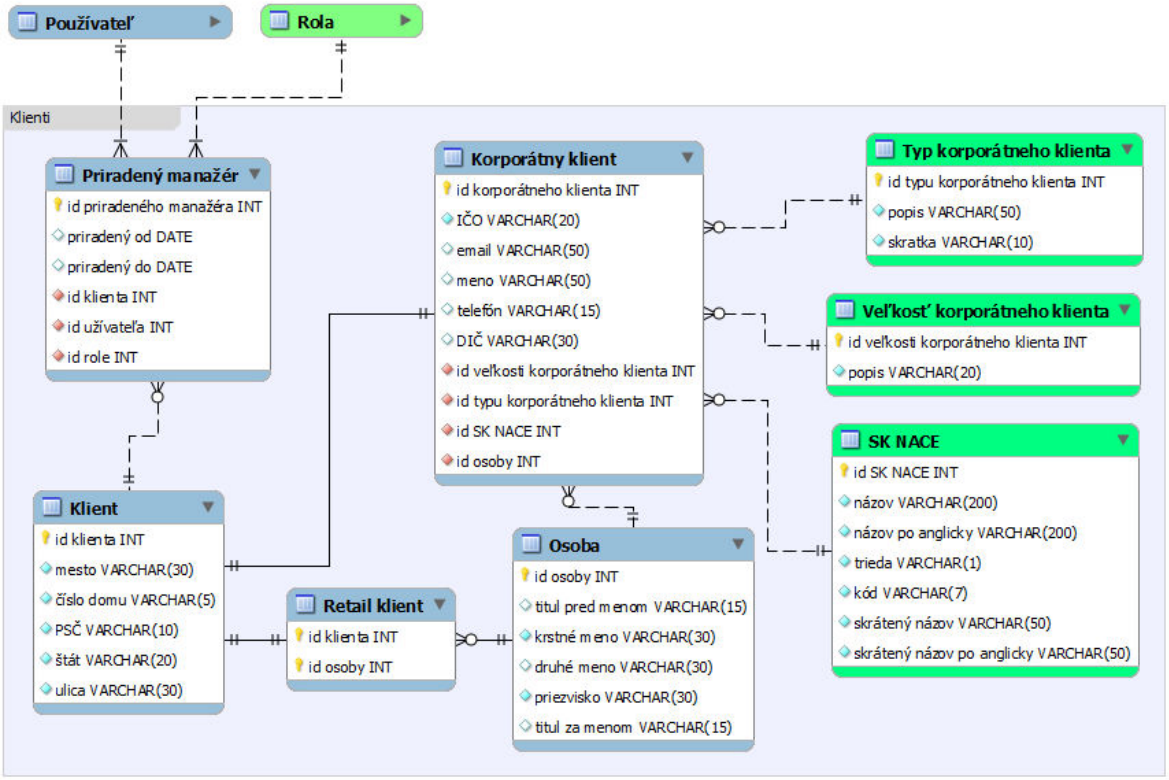
Diagram dokumentov a hodnotení bol z väčšej časti vyvíjaný a implementovaný študentom, Jurajom Branickým, preto uvediem len ich stručnejší popis.

Komponent dokumenty slúži na ukladanie informácií z finančných výkazov. V súčasnom stave má navrhnuté entity pre výkaz ziskov a strát a súvahu. Oba výkazy sa viažu priamo na klienta a sú vytvorené v určitej mene.

Výkaz ziskov a strát tvoria riadky výkazu a meta-dáta o samotnom výkaze. Riadky sú viazané na konkrétny výkaz a dôležitým údajom je ich číslo.

Podobným spôsobom je riešená aj súvaha, kde sú však 2 časti s rôznou štruktúrou - aktíva a pasíva. V EWS sa jednotlivé záznamy vyťahujú na základe čísla riadku.

Obr. 4.4: Dátový model - komponent klienti



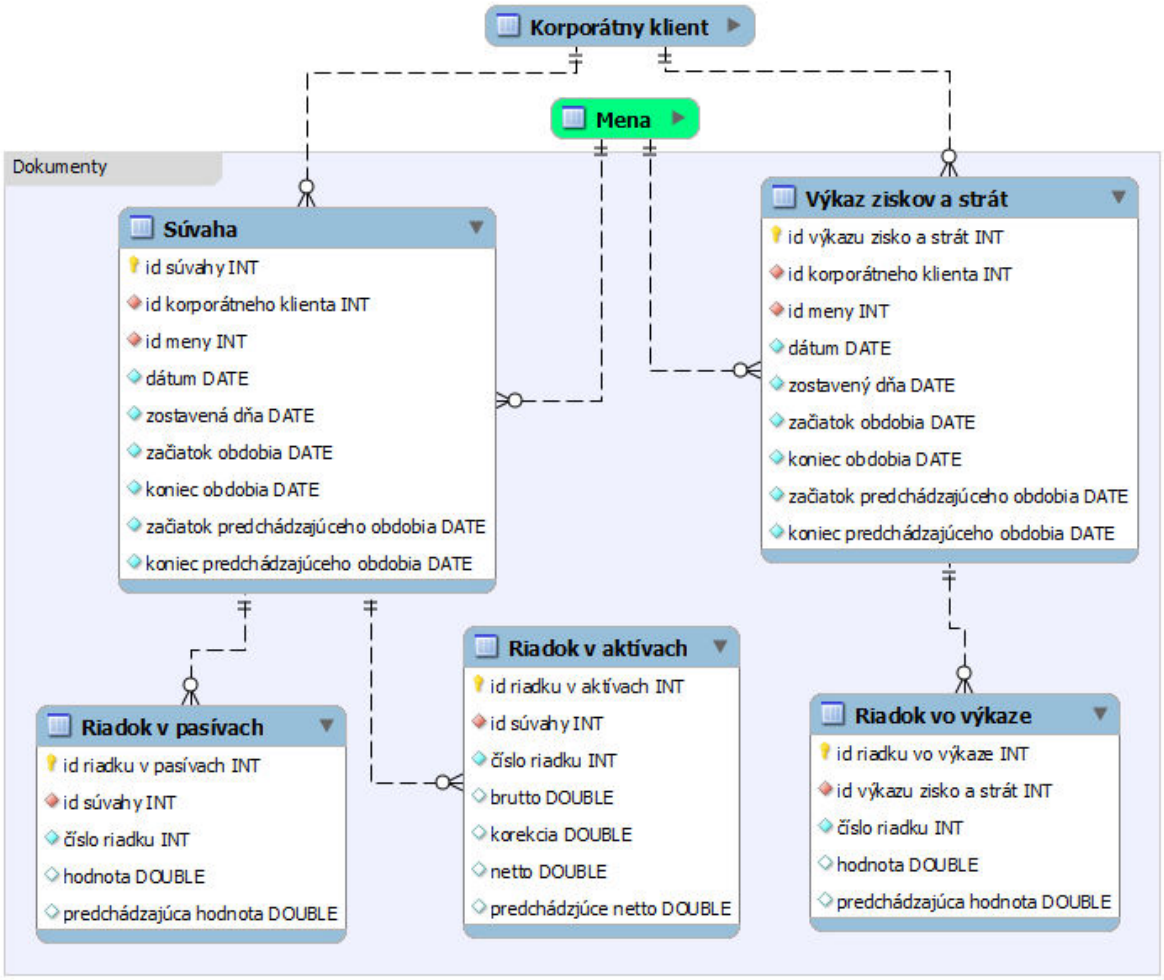
Hodnotenia v sebe zahŕňajú číselníky so všetkými typmi výstrah, ktoré EWS vyhodnocuje, ako aj vážnosť jednotlivých výstrah. Obsah číselníkov je popísaný v kapitole 4.5.

Pri hodnotení klienta sa vytvára záznam v reportoch, ktorý v sebe zahŕňa minimálne 1, zvyčajne niekoľko výstrah. Výstrahy sú naviazané na konkrétne úvery klienta, zatiaľ čo report sa viaže na samotného klienta.

Táto časť komponenty hodnotenie nereprezentuje rozhodnutia používateľov EWS. Tie sú zachytené v entite hodnotenia, ktorá sa skladá z minimálne 1 akcie hodnotenia. Tieto akcie sú definované číselníkmi - povolené akcie sú asociačná entita, ktorá dekomponuje vzťah M:N medzi akciou a rolou.

Takýmto spôsobom sú priradené akcie konkrétnej roli.

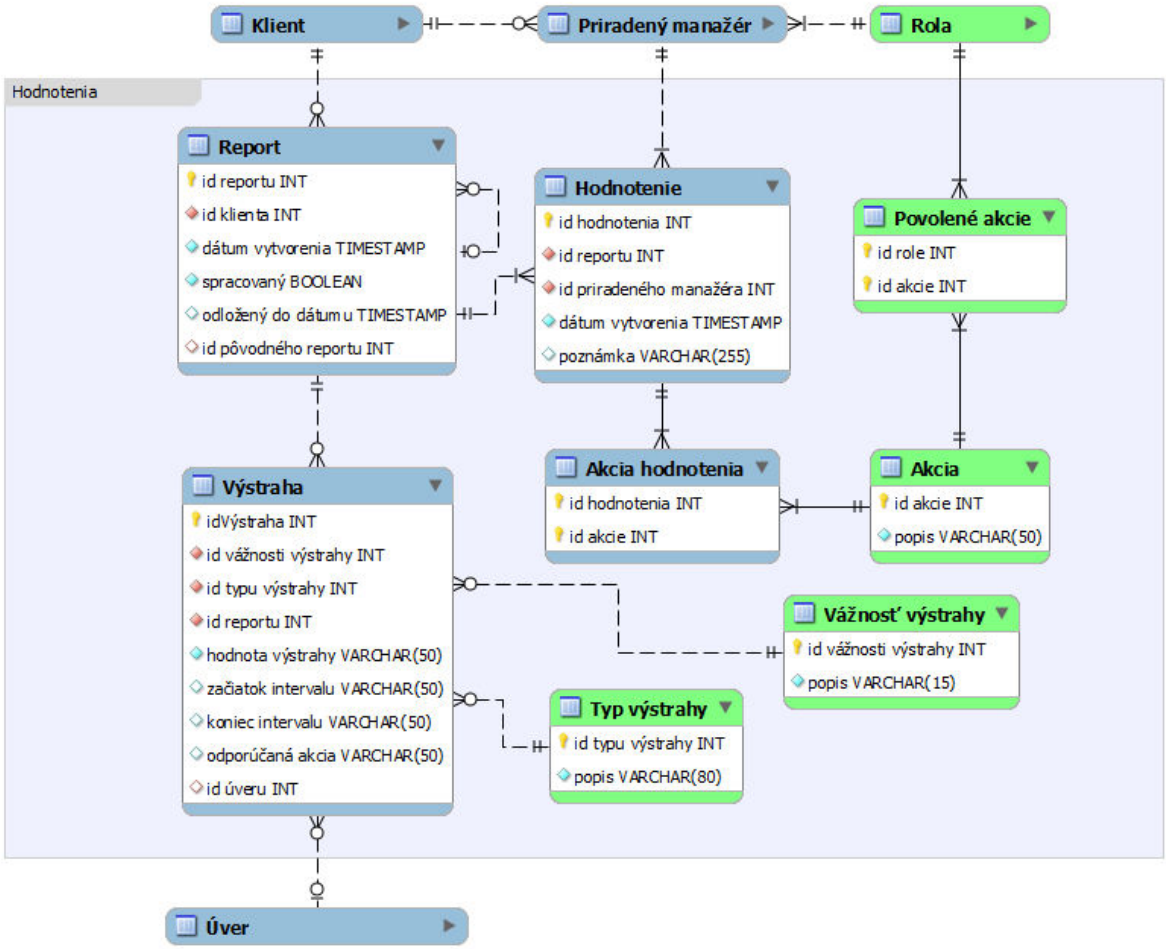
Obr. 4.5: Dátový model - komponent dokumenty



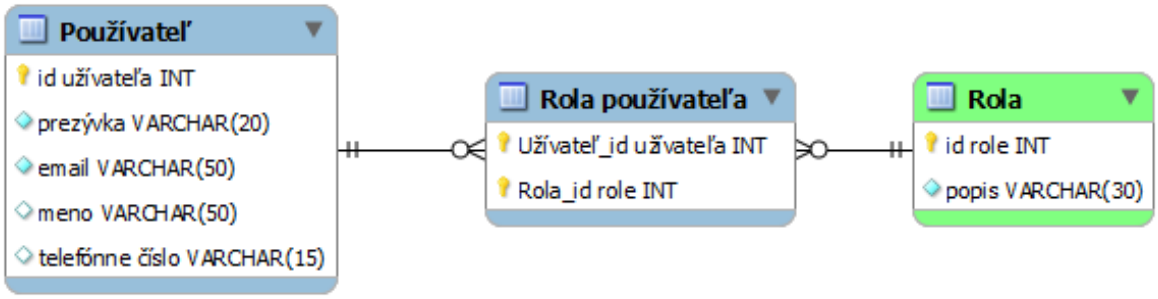
Entita používateľ reprezentuje v našom modeli zamestnanca banky, ktorý využíva alebo prichádza do interakcie s EWS v rôznych roliach. Kvôli dekompozícií vzniknutého vzťahu M:N sme vytvorili asociačnú entitu rolu používateľa.

Používateľ má základné údaje, ktoré slúžia na jeho identifikáciu a kontaktovanie v prípade potreby. Rola je číselník, podľa ktorej sa v komponente hodnotenie určujú možné akcie, ktoré môže používateľ vykonávať.

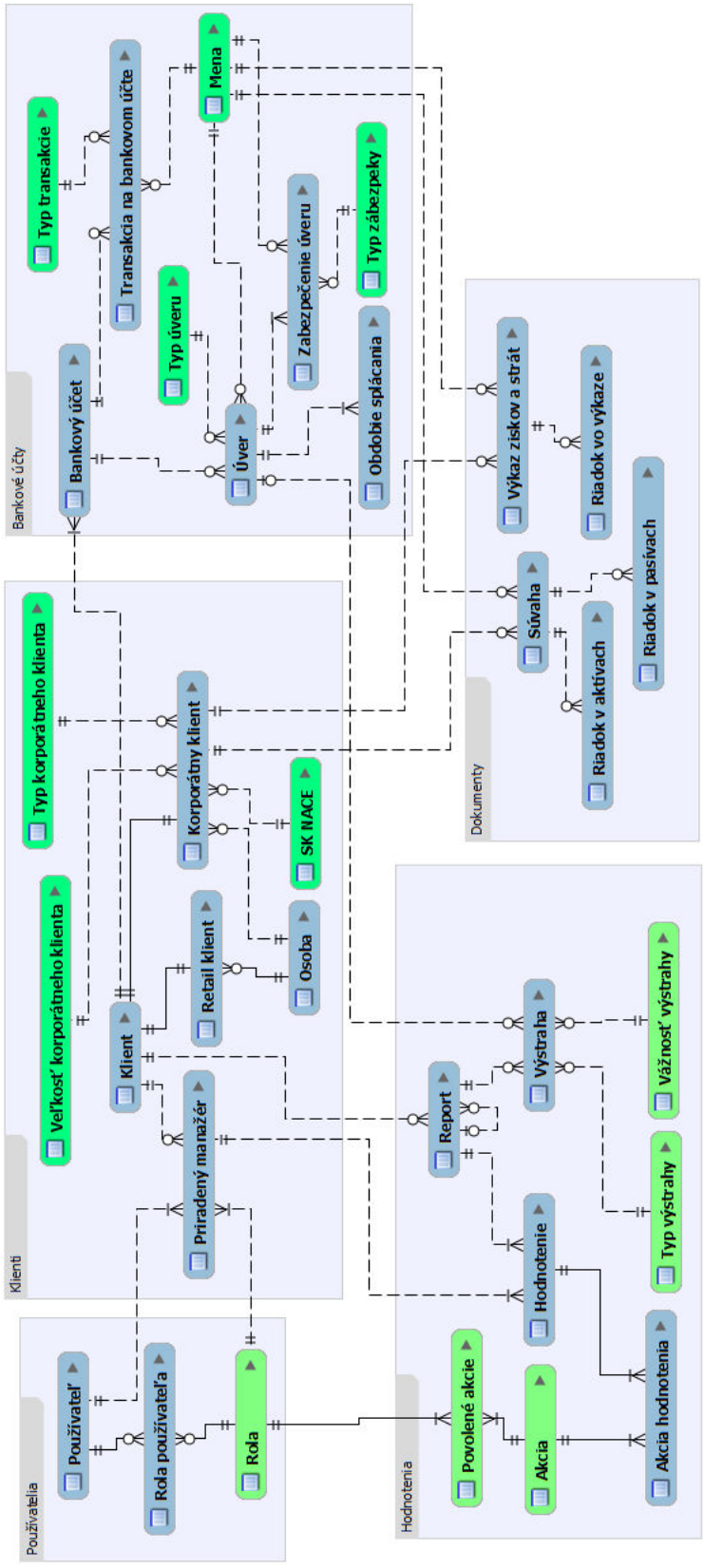
Obr. 4.6: Dátový model - komponent hodnotenia



Obr. 4.7: Dátový model - komponent používatelia



Obr. 4.8: Kompletný dátový model



4.3 Komponentný model riešenia

Komponentný model opisuje spojenie jednotlivých komponentov riešenia, ich zloženie a prepojenie. Je zameraný najmä na väčšie softvérové systémy.

V diagrame EWS sú jasne oddelené komponenty patriace do dátovej, logickej či prezentačnej vrstvy. Dátová vrstva obsahuje niekoľko komponentov, ktoré slúžia ako perzistentné úložisko dát. Predpokladom je vlastná databáza EWS a vlastný dátový sklad. Databáza slúži jednak na uloženie číselníkov, ako aj na logovanie činností z logickej a prezentačnej vrstvy.

Dátový sklad je navrhnutý podľa popisu v kapitole 4.4, vďaka informáciám v ňom uloženým bude možné správne vyhodnotiť zložitejšie obchodné dotazy a robiť hĺbkové analýzy.

Okrem týchto komponentov sú v diagrame uvedené aj externé zdroje. Tými môžu byť vlastné dátové úložiská banky (či už sklad alebo databáza, avšak predpokladom je spracovanie týchto dát do vlastných úložísk), „blacklist“ (zoznam trestne stíhaných osôb, ktorým nie je možné poskytnúť úver), údaje z poisťovní, úradov či iných inštitúcií.

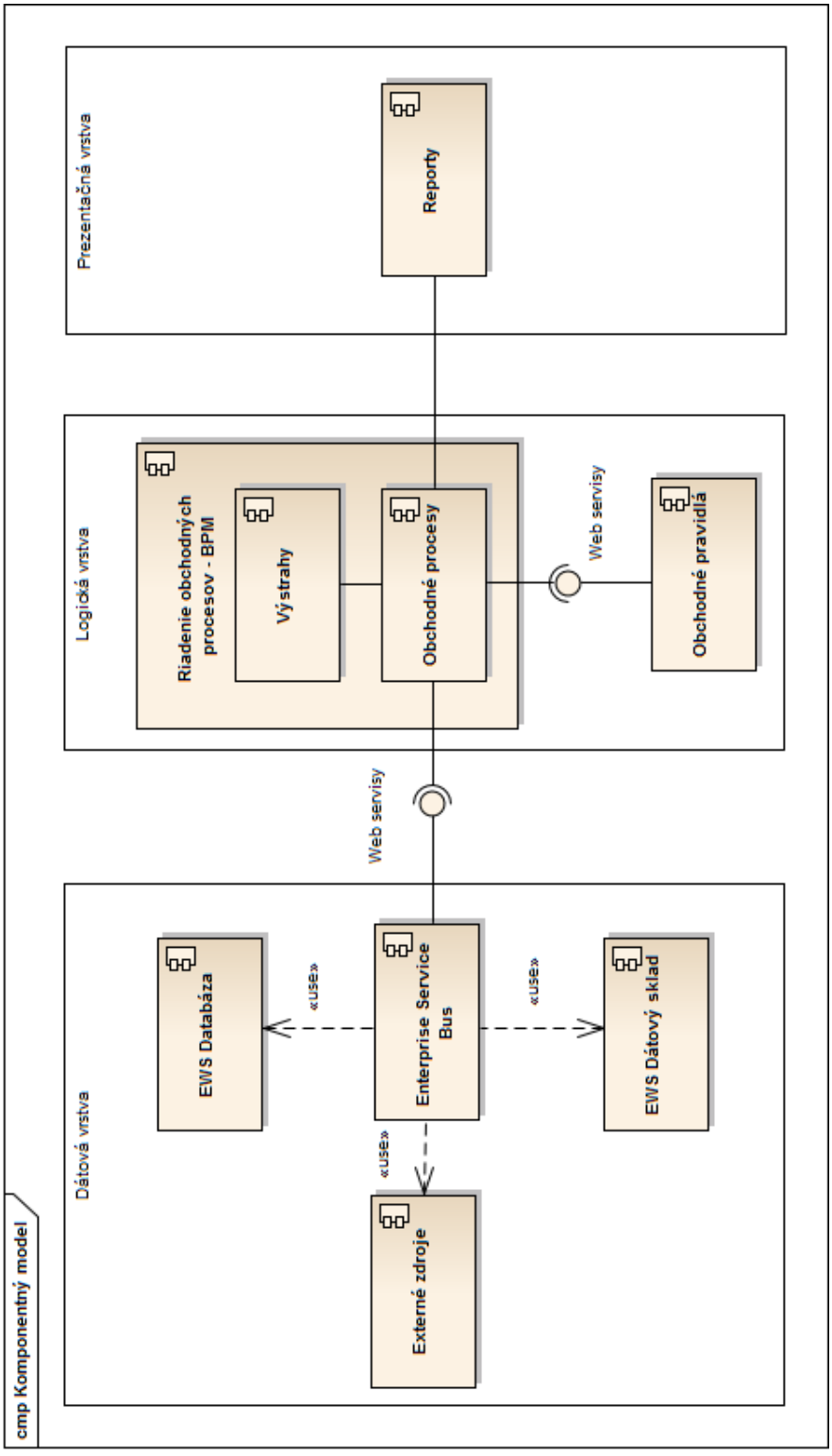
ESB komponent komunikuje so všetkými týmito komponentami, dotazuje sa na ich dáta a vytvára web servisy, ktoré sú prístupné logickej vrstve. Servisy sú volané v rámci obchodných procesov.

Obchodné procesy sú súčasťou systému ich riadenia - BPM. Dôležitým komponentom tohto systému sú výstrahy, ktoré sa v rámci procesov generujú.

V logickej vrstve sa nachádzajú aj obchodné pravidlá, ktoré sú prístupné v podobe servisov a tiež volané v procesoch. Ich definícia a udržiavanie by malo byť nezávislé od BPM systému.

Prezentačná vrstva v EWS závisí taktiež od obchodných procesov a v rámci nich sú generované obrazovky pre EWS používateľov, v našom systéme nazývané aj reporty. Obsahujú aktuálne dáta zo servisov z dátovej vrstvy a výstrahy vyhodnotené pomocou obchodných pravidiel a osobitných servisov.

Obr. 4.9: Komponentný model riešenia



4.4 Návrh dátového skladu

Táto kapitola sa zaoberá stručným návrhom a popisom využitia EWS dátového skladu a jeho významu pre systém ako taký. Hneď na začiatok považujem za nutné poznamenať, že na nám dostupnej platforme nebolo možné dátový sklad fyzicky vytvoriť.
Dátový sklad je navrhovaný za účelom zodpovedania zložitejších obchodných otázok, ktoré majú veľkú informačnú hodnotu, ale je zložité, menej efektívne prípadne aj nemožné ich zodpovedať v klasickej relačnej databáze.
Príklady na obchodné otázky, ktoré by sme chceli zodpovedať v EWS sú:

- Koľko výstrah určitého typu bolo vygenerovaných u konkrétneho klienta za určité časové obdobie?
- Aký druh výstrahy je najpočetnejší pre určité odvetvie podnikania?
- Aký je počet kreditov, debetov, splátok, vykonaných z účtu klienta za dané časové obdobie?
- Aká je suma kreditov, debetov, splátok, vykonaných z účtu klienta za dané časové obdobie?
- Koľkým klientom v danom odvetví a danej veľkosti boli znížené limity čerpania úveru? Prípadne vyžiadaná dostatočná zábezpeka?

Jednotlivé otázky si vyžadujú zložité spájanie tabuliek, vyhodnocovanie dotazov, ktoré by boli pomerne časté, avšak v relačnej databáze pomalé a náročné. Špecializovaný EWS dátový sklad by bol určený na riešenie práve tohto problému.
Devízou by bolo ukladanie komplexných logovacích dát - histórie riešenia klienta v EWS. Táto história by mohla priniesť veľké množstvo dodatočných indícií na posúdenie jeho situácie a stať sa tak veľkou konkurenčnou výhodou nami navrhovaného EWS.

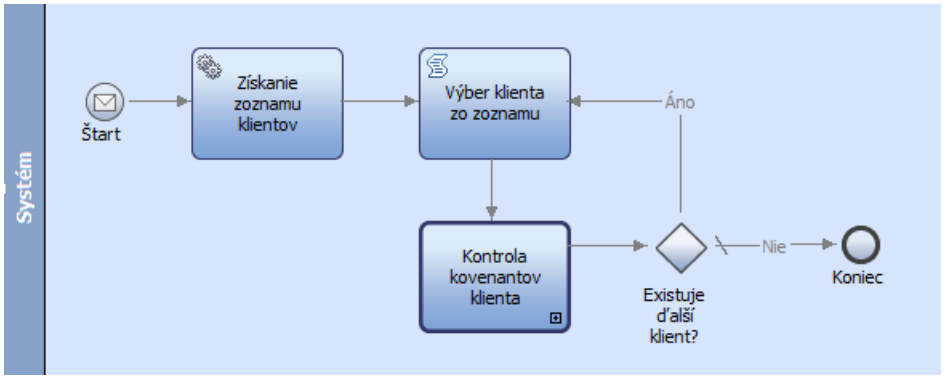
4.5 Popis systému a jeho súčastí

V tejto časti rozpíšem hlavnú časť mojej práce, návrh dôležitých číselníkov, logiku aplikácie, základné procesy, jednotlivé servisy vytvorené v nástroji Message Broker, koncept a dátový model EWS a iné.

Servisy, ktoré poskytujú dáta z dátovej vrstvy, sú volané a používané v rámci obchodnej vrstvy - BPM. Táto vrstva má v sebe celú logiku aplikácie a zaobstaráva jednotlivé pravidelné i mimoriadne kontroly klientov. Pre lepšie pochopenie preto uvádzam aj workflow samotných procesov v BPM.

Štartovací proces využíva servisy s informáciami o klientoch, zavolá si ich zoznam a postupne pre každého z nich volá kontrolu kovenantov, ktorá je rozpísaná nižšie.

Obr. 4.10: Štartovací proces - spúšťanie systému



Pri kontrole kovenantov konkrétneho klienta sa vždy skontrolujú pohyby na účte, pretože tie sa menia najčastejšie a je potrebné ich overiť v čo najmenších intervaloch. Tak sa v prípade alertu môžu vykonať protiopatrenia s minimálnym časovým sklzom.

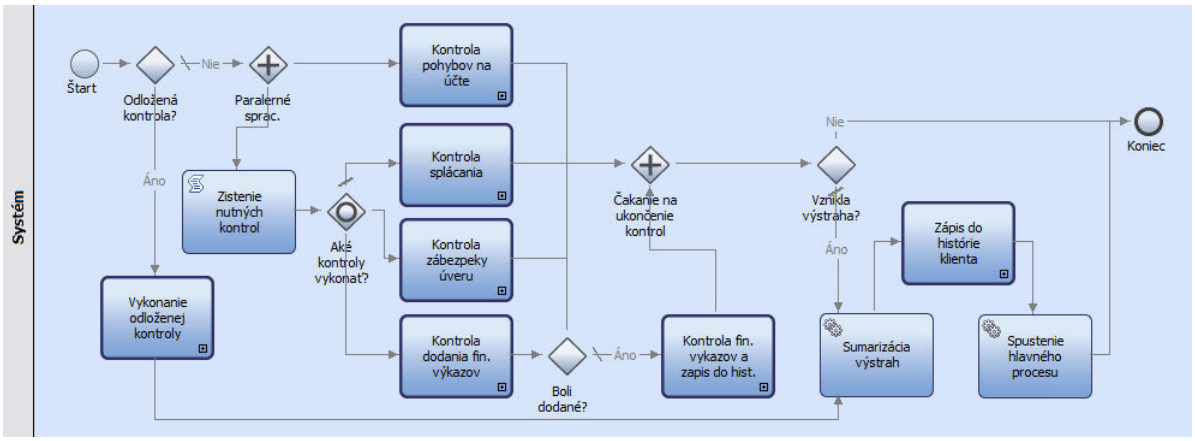
Ďalšie kontroly sa nevykonávajú vždy, najskôr sa zistí, ktoré z nich je nutné vykonať - dá sa to určiť na základe času, periodicity a iných faktorov. Jedná sa o kontrolu splácania úveru (periodicita je určená zmluvne podľa obdobia splácania, zvyčajne mesačne), kontrola zábezpeky úveru (kontrola prebieha jedenkrát, prípadne pri vyžiadaní dodatočnej zábezpeky) a kontrola dodania finančných výkazov (jedenkrát ročne, prípadne s inou periódou).

Ak sú finančné výkazy dodané, dá sa na nich vykonať kontrola, z ktorej môžu vyplynúť alerty číslované od 400. Alerty číslované od 100 sa týkajú kontroly transakcií, alerty číslované od

200 kontroly splátok a alerty číslované od 300 dodania finančných výkazov.

Kontrola dát z finančných výkazov je najzložitejšia a je základom predikcie správania sa klienta. Je vykonávaná v IBM WebSphere ILOG.

Obr. 4.11: Proces kontroly konkrétneho klienta



Systém kontroly je založený na alertoch, ktoré môžu byť vyvolané pri posudzovaní klienta. Zoznam alertov (výstrah) navrhnutých a používaných v systéme:

- Skupina alertov číslovaná od 100 sa týka transakčných dát:
  - **101** - Nezvyčajne vysoká transakcia
  - **102** - Transakcia na podozrivý účet (mimo EU)
  - **103** - Výrazná zmena zostatkov
- Skupina alertov číslovaná od 200 sa týka splátok úveru:
  - **201** - Nebola zaplatená splátka
  - **202** - Bola zaplatená iba časť splátky
- Skupina alertov číslovaná od 300 sa týka finančných výkazov (najmä súvaha a výkaz ziskov a strát):
  - **301** - Nebol dodaný finančný výkaz
  - **302** - Neboli dodané dokumenty záložného práva
- Skupina alertov číslovaná od 400 sa týka faktov zistených na základe kontroly výkazov:
  - **401** - Výstraha z kontroly finančného výkazu VOĽNÉ ZDROJE - EBITDA
  - **402** - Výstraha z kontroly finančného výkazu RENTABILITA - aktív čistá
  - **403** - Výstraha z kontroly finančného výkazu RENTABILITA - vlastného kapitálu
  - **404** - Výstraha z kontroly finančného výkazu LIKVIDITA - celková
  - **405** - Výstraha z kontroly finančného výkazu ZADĹŽENOSŤ - aktív, celková
  - **406** - Výstraha z kontroly finančného výkazu ZADĹŽENOSŤ - Úroveň samofinancovania
  - **407** - Výstraha z kontroly finančného výkazu ZADĹŽENOSŤ - Úrokové krytie
  - **408** - Výstraha z kontroly finančného výkazu AKTIVITA - Doba obratu zásob z nákladov

- **409** - Výstraha z kontroly finančného výkazu AKTIVITA - Doba obratu pohľadávok
- **410** - Výstraha z kontroly finančného výkazu AKTIVITA - Doba obratu záväzkov
- **411** - Výstraha z kontroly finančného výkazu Objem neobežného majetku
- **412** - Výstraha z kontroly finančného výkazu Obežný majetok - Krytie
- **413** - Výstraha z kontroly finančného výkazu PRACOVNÝ KAPITÁL
- **414** - Výstraha z kontroly finančného výkazu - Celkové KPI

Každý z uvedených alertov je posudzovaný v IBM WebSphere ILOG a môže nadobudnúť nasledovné stavy vážnosti:

- **OK** - kontrola prebehla v poriadku a skúmaný jav je v medziach tolerancie.
- **NOK** - kontrola ukázala na anomáliu, ktorá však nebola závažná. Posúdenie je potom na EWS analytikovi.
- **ALERT** - kontrola zistila výrazné prekročenie hraníc tolerancie skúmaného javu a je nutné tento problém riešiť.

Alerty posudzujú užívatelia EWS, ktorí majú rôzne role s rôznymi právami. Jednotlivé navrhované role a ich právomoci a úlohy sú:

1. **EWS analytik** - ak kontrola zistí určitú anomáliu pri klientovi, EWS analytik je prvý, kto sa dostane k posudzovaniu tejto výstrahy, pokiaľ nie je vážnejšia (napríklad pri nedodaní finančných výkazov - 301 a 302). Analytik môže daný alert uzatvoriť, s tým, že bol neopodstatnený, alebo sa vyriešil, alebo ho eskalovať ďalej na vzťahového manažéra.
2. **Vzťahový manažér** - je v hierarchií riešenia problémov hneď za analytikom. Dostávajú sa k nemu závažnejšie problémy a v jeho právomoci je okrem uzavretia aj eskalácia na risk manažéra či odloženie alertu na určitú dobu.

3. **Risk manažér** - má možnosť ukončiť prípad, odložiť ho, eskalovať na WatchList alebo Workout manažéra, alebo navrhnúť určitú dodatočnú akciu (akcie 1 až 5) - viď tabuľka nižšie. Eskalovaním na WatchList manažéra sa klient dostáva do WatchListu.

4. **WatchList manažér** - monitoruje klientov vo WatchListe a má možnosť uzatvoriť daný prípad, presunúť ho z WatchListu na vzťahového, risk alebo Workout manažéra alebo navrhnúť dodatočnú akciu (akcie 1 až 5).

5. **Workout manažér** - vykonáva výhradne odpis pohľadávky - akciu č. 6.

Obr. 4.12: Právomoci jednotlivých rolí

Právomoc\Rola	EWS analytik	Vzťahový manažér	Risk manažér	Watchlist manažér	Workout manažér	System
Potvrdenie alertu	X					
Posunutie alertu	X	X	X	X		
Kontakovanie klienta		X			X	
Úprava ratingu			X	X		
Zaradenie do watchlistu			X			
Potvrdenie pridania do watchlistu				X		
Spustienie kontroly splácania úveru		X	X	X		X
Spustenie kontroly pohybov na bežnom účte			X	X		X
Plánovanie opakovanej kontroly		X	X			
Postúpenie klienta na workout			X	X		
Zrušenie klienta z workout					X	
Spúšťanie kontroly klientov						X
Počiatočné rozhodnutie o priradení alertu roli						X

Dodatočné akcie, ktoré môžu vykonať manažéri sú uvedené v kapitole 3.3.

4.6 Implementované servisy

Jednotlivé rozdelenia a fakty vyplývajú z analýzy a sú dohodnuté celým tímom, ktorý implementuje EWS. Sú zachytené v podobe dát v nami implementovanej databáze a statické dáta sú uložené v podobe číselníkov, ktoré sú jednotné pre všetky vrstvy.
Gro práce v IBM WebSphere Message Broker nespočíva iba v práci s databázou, ale najmä v poskytovaní potrebných dát v podobe obchodných servisov. Tieto servisy poskytujú dáta obchodnej vrstve (v našom prípade IBM WebSphere BPM), ktorá potrebuje špecifické dáta, buď na zobrazenie v samotnom EWS, alebo ako dáta pre posudzovanie v systéme IBM WebSphere ILOG.
Architektúra systémov je podľa princípov SOA popísaných v kapitole 1.2. Hlavná výhoda je zastrešenie práce s databázou a z toho vyplývajúca nezávislosť obchodnej vrstvy od databázového modelu či jeho konkrétnej implementácie.
Zoznam servisov, aktívne využívaných v systéme a ich účel:

1. **Informácie o klientovi** - základné informácie o klientovi na informatívne zobrazenie.
   Operácie:

   - Vráť informácie o klientovi na základe jeho jedinečného ID,
   - Vráť informácie o klientovi na základe jeho IČO identifikátoru,
   - Vráť zoznam klientských ID - slúži na spustenie pravidelnej kontroly jednotlivých klientov, zoznam bude obsahovať klientov, ktorí majú úver a je potrebné ich skontrolovať. Okrem samotných ID sa pre každého klienta uvádza aj dátum, od ktorého má platný úver - slúži na vyrátanie splátok a ich overenie.

   Zoznam vrátených položiek:

   - Názov
   - Adresa
   - IČO
   - Typ spoločnosti

- Telefónne číslo
- Email
- SK NACE (predmet prevažujúcej činnosti)

2. **Priradení manažéri** - vráti manažérov, ktorí sú v danom čase priradení danému klientovi. Slúži na identifikáciu osoby zodpovednej za posúdenie klienta. Keďže sa priradený manažér môže časom zmeniť, je možné nájsť túto osobu aj historicky v danom časovom okamihu.
   Operácie:
   - Vráť priradených manažérov pre daného klienta na základe jeho ID,
   - Vráť priradených manažérov pre daného klienta na základe jeho IČO identifikátoru.

   Zoznam vrátených položiek:
   - Meno
   - Priezvisko
   - Typ
   - Telefónne číslo
   - Email
   - Nick - pred identifikáciu v systéme (ako sa používateľ prihlasuje)
3. **Finančné výkazy** - slúži na kontrolu posledne odovzdanej súvahy / výkazu ziskov a strát. Kontrola bude vykonávaná v nástroji ILOG.
   Operácie:
   - Vráť informácie o najnovšie odovzdanej súvahe podľa ID klienta,
   - Vráť informácie o najnovšie odovzdanom výkaze ziskov a strát podľa ID klienta.

   Zoznam vrátených položiek:
   - Platnosť - dátum od - údaj z finančného výkazu

- Platnosť - dátum do - údaj z finančného výkazu
- Reálny dátum odovzdania - - údaj z finančného výkazu
- Perióda - s ktorou má byť výkaz odovzdávaný

4. **Agregáty transakcií** - slúži na zobrazenie agregovaných informácií o transakciach klienta.

   Operácie:

   - Vráť agregáty transakcií na základe ID klienta.

   Zoznam vrátených položiek:

   - Suma kreditov z účtov klienta
   - Suma debetov z účtov klienta
   - Suma debetov do sociálnej poisťovne
   - Počet kreditných transakcií
   - Počet debetných transakcií
   - Počet protiúčtov, na ktoré smerovali debety z účtu

5. **Zabezpečenie úveru** - odosiela informácie o zábezpekách za úvery klienta.

   Operácie:

   - Vráť zábezpeky na základe ID klienta.

   Zoznam vrátených položiek:

   - Typ zabezpečenia úveru - hnuteľný, nehnuteľný a iné
   - Názov - o akú zábezpeku sa presne jedná
   - Hodnota zábezpeky - v mene
   - Platnosť od - dátum, od ktorého je zábezpeka platná
   - Platnosť do - dátum, do ktorého je zábezpeka platná
   - Dátum odovzdania - kedy bola zábezpeka odovzdaná

6. **Informácie o úveroch** - odosiela kompletné informácie o klientských úveroch.
   Operácie:

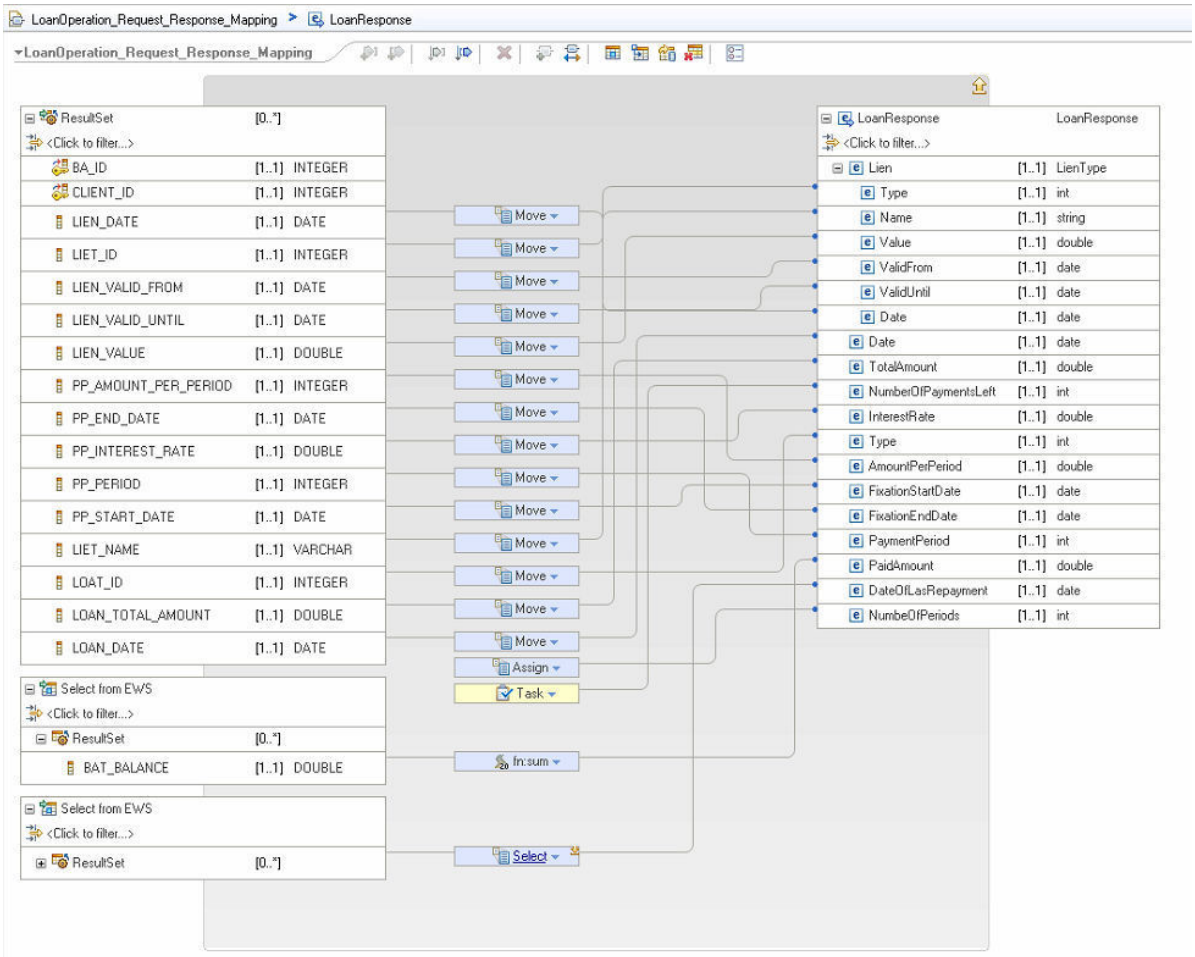
   - Vráť informácie o úveroch klienta s daným ID.

   Zoznam vrátených položiek:

   - Informácie o záložnom práve - štruktúra ako pri predchádzajúcom servise
   - Dátum podpisu
   - Schválený objem - celková požičaná suma
   - Počet zostávajúcich splátok - vypočíta sa podľa periódy splácania v danom období
   - Úroková sadzba - platí pre konkrétne obdobie splácania
   - Typ úveru - podľa číselníka, viď dátový model
   - Aktuálna výška splátky - podľa konkrétneho obdobia splácania
   - Začiatok fixácie úveru - dátum - podľa obdobia splácania
   - Koniec fixácie úveru - dátum - podľa obdobia splácania
   - Perióda splácania - z príslušného obdobia splácania, zvyčajne 3 mesiace
   - Doteraz zaplatený objem - určí sa na základe príslušných transakcií
   - Dátum poslednej splátky - určí sa na základe príslušných transakcií
   - Počet periód - počet periód do splatenia úveru

   Servis bol implementovaný podľa postupu uvedeného v kapitole 2. Projekt vo WebSphere Message Broker Toolkit využíva databázovú definíciu, cez ktorú je možný prístup k vytvorenej databáze.
   Na transformáciu vstupnej správy, ktorá obsahuje parameter - ID klienta - na správu výstupnú, ktorá obsahuje všetky uvedené polia, sa využíva už spomínané grafické mapovanie aj s databázovými operáciami.
   Mapovanie použité v servise je uvedené na obrázku 6.

Obr. 4.13: Mapovanie - získanie údajov pre informácie o úvere



7. **Druhy akcií** - odosiela zoznam všetkých dostupných akcií, ktoré je možné vykonať pri posudzovaní jednotlivých výstrah.
   Operácie:

   - Vráť všetky dostupné akcie - bez parametrov.

   Zoznam vrátených položiek:

   - ID akcie - jednoznačný identifikátor v rámci databázy, číselník
   - Popis akcie - názov akcie, pre prezentačné účely

8. **Prehľad transakcií** - sprístupňuje celkový zoznam transakcií vykonaných na účtoch klienta s daným ID a od daného času.
   Operácie:

   - Vráť všetky transakcie na základe ID klienta a dátumu od.

   Zoznam vrátených položiek:

   - Suma v lokálnej mene - v našom prípade Eurá
   - Zostatok na účte - po vykonaní transakcie
   - Protiúčet, na ktorý smerovala transakcia
   - Dátum, kedy transakcia prebehla
   - Variabilný symbol transakcie
   - Typ transakcie (debet, kredit, splátka)

9. **EWS história klienta** Servis bol implementačnou zodpovednosťou študenta Juraja Branického, a preto ho nebudem v mojej práci presnejšie popisovať. Definované operácie sú:

   - Vráť históriu s parametrom filtruj iba alerty - vráti zoznam všetkých druhov alertov, ktoré sa týkajú klienta. V prípade filtrovania sa vypíšu iba alerty, ktorý stav je ALERT (OK a NOK sa ignorujú).

- Zaloguj dané historické dáta - vstupom je ID klienta a zoznam alertov aktuálneho reportu, ktoré treba zalogovať. V rámci spracovania servisu sa vytvorí záznam v tabuľkách report a spravidla niekoľko záznamov v tabuľke alert.

10. **ILOG dátové servisy** Servis s niekoľkými operáciami poskytuje potrebné dáta pre vyhodnocovanie v systéme ILOG. Servisy však prevoláva a manažuje BPM. Operácií a ich atribútov je väčšie množstvo, ich popis bude aj vzhľadom na podobný princíp s predchádzajúcimi servismi stručnejší.
    Operácie:

    - Základné informácie o klientovi - odoslanie typu a veľkosti spoločnosti pre konkrétne ID klienta.
    - Zmluvné podmienky - určenie, či bola dodaná súvaha a výkaz ziskov a strát spolu s typom zábezpeky, ktorou sa ručí za úver. Parametrom je ID klienta.
    - Aktuálne informácie o úvere - odoslanie výšky splátky, celkovej dlžnej sumy, požadovanej sumy, celkového počtu splátok a počtu zaplatených splátok.
    - Informácie o úvere - odoslanie podrobnejších informácií o úvere, vrátane dňa splácania, periódy, dátumu poslednej splátky, fixácie, splácania, jeho výšky a iné.
    - Finančné výkazy - odoslanie jednotlivých riadkov súvahy a výkazu ziskov a strát evidovaných pre klienta.
    - Zoznam pohybov na účte - usporiadaný zoznam klientských transakcií spolu s podrobnejšími údajmi - slúži na vyhodnotenie výstrah.

Kapitola 5

Ukážka systému včasného varovania a jeho využitie

5.1 Prezentačná vrstva EWS

Finálny EWS je zabezpečený systém, ktorý je určený pre EWS používateľov. Pre každú rolu je obrazovka prispôsobená v zmysle povolených akcií a zobrazovaných informácií. Systém má podobu webového portálu a je tak univerzálny a platformovo nezávislý. Screenshot úvodnej informatívnej obrazovky je možné vidieť na obrázku 5.1.
Podoba tohto portálu bola tvorená priamo v BPM, v nástroji Process Designer, pomocou tzv. Coachov. Coach slúži na definovanie formulárových výstupov z jednotlivých procesov, popísaných vyššie. Pre prehľadnejšie a bohatšie užívateľské rozhranie boli jednotlivé formuláre rozšírené aj o nadštandardné prvky.
Formulár má niekoľko častí:

- **Časť klient** je informatívna a zobrazuje informácie o korporátnom klientovi- údaje zo servisov: Informácie o klientovi, priradení manažéri a taktiež ukazovatele do grafu. Tento radarový graf je dôležitým vizuálnym prvkom a určuje situáciu klienta - zeleným polygónom je zobrazená ideálna situácia podľa dát z ILOG-u, červeným polygónom skutočná klientova situácia. Bodmi polygónu sú metriky, popísané pod grafom. V našom systéme boli vybraté tieto:

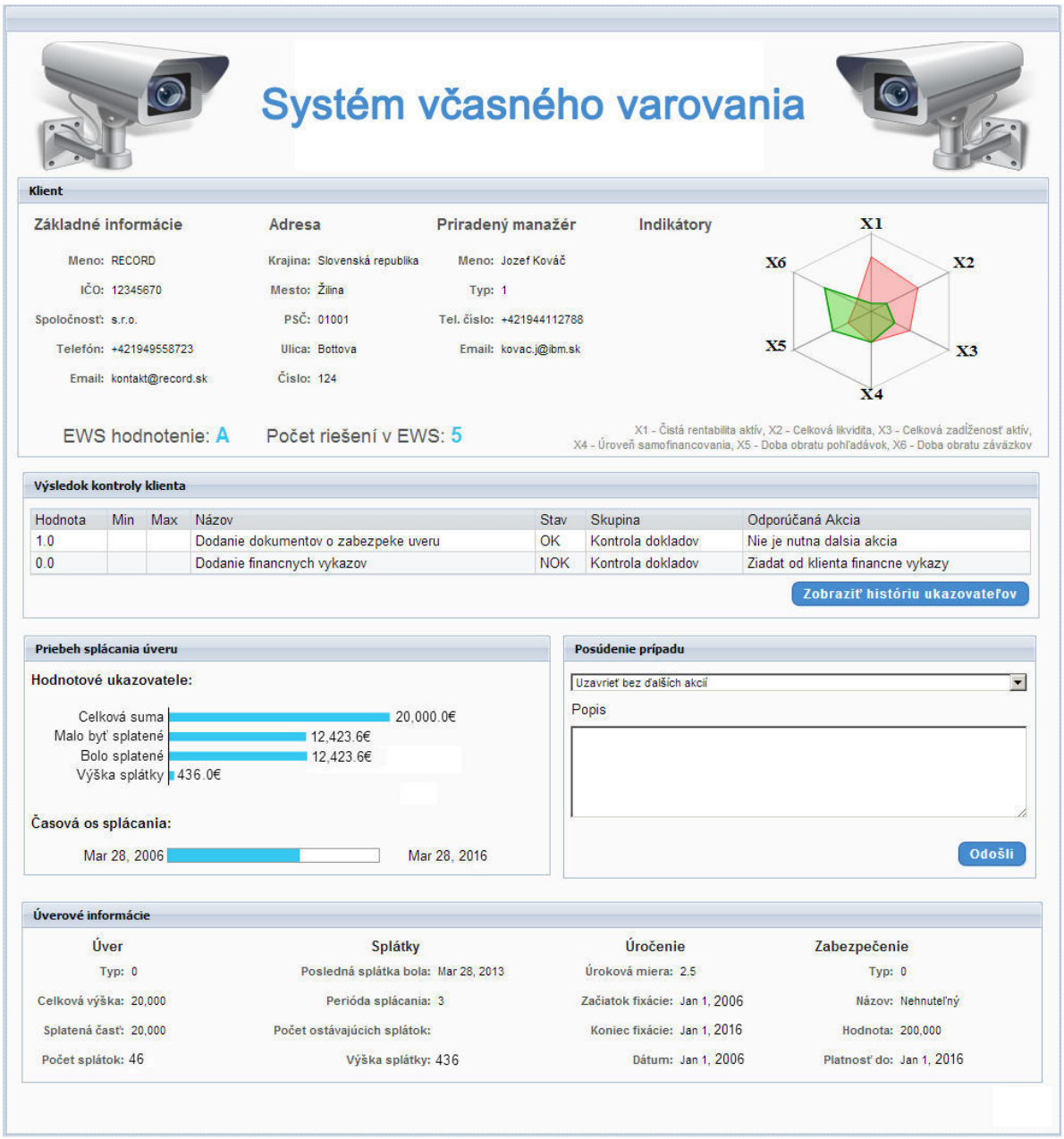
  - **X1** - Čistá rentabilita aktív
  - **X2** - Celková likvidita aktív
  - **X3** - Celková zadĺženosť
  - **X4** - Úroveň samofinancovania
  - **X5** - Doba obratu pohľadávok
  - **X6** - Doba obratu záväzkov

  Podrobnejší popis jednotlivých metrík a spôsobu ich určenia neuvádzam, keďže obchodné pravidlá EWS neboli predmetom mojej diplomovej práce.

- **Časť výsledok kontroly klienta** je zobrazenie výsledkov poslednej kontroly, prevolanie servisu EWS história klienta. Umožňuje i úplný prehľad všetkých kontrol.
- **Časť priebeh splácania úveru** zobrazuje súhrnný prehľad splácania úveru, využíva dáta zo servisu informácie o úvere.
- **Časť posúdenie prípadu** umožňuje posúdiť daný prípad, využíva servis druhy akcií. Zobrazuje rôzne výsledky podľa role používateľa.
- **Časť úverové informácie** zobrazuje úverový prehľad so zábezpekou a splátkami zo servisu informácie o úvere.

Z uvedených opisov je zjavné prepojenie logickej, dátovej a prezentačnej vrstvy pomocou servisov. Jednotlivé akcie EWS používateľov sa taktiež logujú do databázy a EWS história klienta sa aktualizuje.

Obr. 5.1: Ukážka EWS - prehľad korporátneho klienta



Na obrázku 5.1 je zobrazená história riešenia klienta - konkrétne 1 kontrola (report) aj s hodnotami pre jednotlivé výstrahy. Nevyplnené výstrahy neboli počas kontroly vygenerované. Report zobrazuje aj príslušného priradeného manažéra a dátum vytvorenia.

Obr. 5.2: Ukážka EWS - prehľad histórie klienta - výstrahy

Detaily Reportu

	Kód výstrahy	Hodnota	Povolené min.	Povolené max.	Status	Odporúčaná akcia
○	101	11,23%		10%	NOK	Preverenie klienta
○	102				OK	
○	103	5,2%		10%	OK	
○	201	1			OK	
○	202	1			OK	
○	301	1			OK	
○	302	1			OK	
○	401	-133,05%	5%		ALERT	
○	404	0,94	1.6		ALERT	
○	402	-48,7%	15%		ALERT	
○	405	106%		50%	ALERT	
○	406	-5,62%		30%	ALERT	
○	409	98,41 dní		40 dní	ALERT	
○	410	94,81dní		60 dní	NOK	
○	411					
○	403					
○	413					
○	412					
○	414					
○	407					
○	408					

Vytvorené	Manažér	Rola	Poznámka
15.4.2013	Jozef Kováč	Relationship	Klient posnutý na risk manažment

5.2 Testovanie vzniknutého riešenia

Testovanie riešenia je dôležitá a najmä časovo náročná fáza. Kód vyprodukovaný programátorom má tendenciu obsahovať bugy, ktorých pravdepodobnosť výskytu ešte zvyšuje obchodná logika a zložitosť systému.
V našom prípade by sa dali možné testy kategorizovať nasledovne:

- Testy servisov - dátových, ILOG a iných (unit testovanie)
- Integračné testy - fungovanie prepojenia celého systému (integrácia a testovanie)
- Test správnosti EWS - vytvorenie prípadov aj s predpokladaným výsledkom a test správnosti ich vyhodnotenia (validácia)

Testy rozoberiem podrobnejšie v ďalších kapitolách. Čo sa týka testu správnosti EWS, predpokladom na jeho vykonanie je značné množstvo dát a netriviálna príprava, ktorá bola v súčasnom stave systému implementovaná len v obmedzenej miere.

5.2.1 Testy servisov

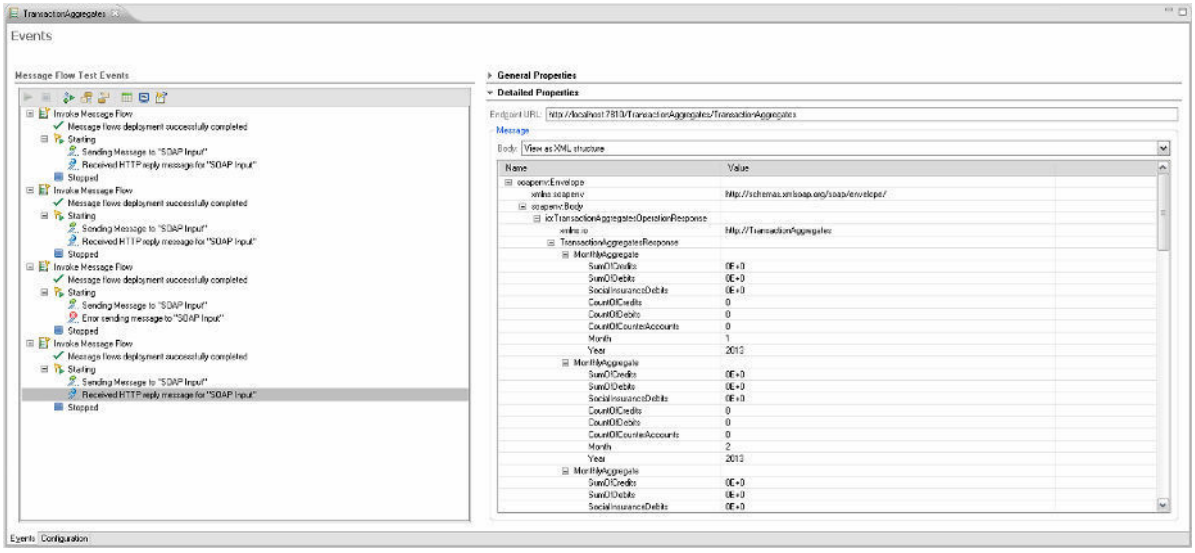
Na testovanie servisov sa využívali technológie na platforme IBM WebSphere Message Broker, ako aj voľne dostupné nástroje - nettool a soapUI. Pre testovacie účely boli vytvorené viaceré skupiny vykonávania, na ktoré sa jednotlivé projekty zaobaľujúce servis nasadzujú.
Každý servis má vlastnú skupinu vykonávania a skupina default sa využíva na nezávislé testovanie. Servis môže byť nasadený v rôznych skupinách. Takýmto spôsobom modifikácia / nefunkčnosť jedného servisu nezmení žiadny iný, už nasadený servis.
Test v nástroji IBM WebSphere Message Broker Toolkit prebieha v špeciálnom prostredí, ktoré je znázornené na obrázku 5.2.1. Nástroj podľa definície servisu zistí všetky operácie aj s formátom dát, ktorý je očakávaný na vstupe. Vstupnú správu pre zvolenú operáciu je ďalej možné ľubovolne modifikovať.
Pri prvom spustení sa vyberie skupina vykonávania a test sa na ňu nasadí, proces trvá dlhšie, každé ďalšie volanie je potom rýchlejšie. Pri zmene závislých prvkov sa test znovu načíta a

nasadí. Uchovávajú sa predtým odoslané správy aj ich výsledky, je možnosť krokovať vykonávanie toku správ a zistiť tak chybu.

Nevýhodou prostredia je zatiaľ nedokončená implementácia nového prvku - grafických máp, v ktorých krokovanie zatiaľ nefunguje a taktiež horšie čitateľný formát výnimok. Napriek týmto skutočnostiam je tento nástroj dostatočný pre testovanie servisov pred ich ostrým nasadením.

V logickej vrstve, ktorá sa môže fyzicky nachádzať na inom serveri, nie je možné testovať týmto spôsobom. V tom prípade sa využívajú vyššie spomenuté voľne dostupné nástroje, kde sa dá overiť funkčnosť servisov alebo ich očakávané vstupy.

Obr. 5.3: Testovanie servisov - agregáty transakcií



5.2.2 Integračné testy

Integračné testovanie nasleduje logicky po unit testoch a predchádza validácií riešenia. Ak fungujú samostatne jednotlivé servisy, je potrebné otestovať EWS, kde budú servisy volané a využívané.

Systém v našom prípade pozostáva okrem dátovej vrstvy aj z osobitných obchodných pravidiel v logickej vrstve, ktoré sú taktiež volané ako servisy. To robí integračné testy o to

zložitejšími.
Test prebiehal najskôr v minimalistickej podobe, kde sa volal servis z dátovej vrstvy a obchodný servis a údaje z oboch sa potom zobrazili v prezentačnej vrstve.
Pre správne fungovanie bolo treba riešiť aj sieťové problémy, keďže jednotlivé nástroje boli umiestnené na osobitných serveroch v Bratislave a v rámci tímu sa k nim pristupovalo cez vzdialenú plochu, prípadne nástroj TeamViewer.
Testy prebiehali za účasti celého tímu a po overení funkčnosti základného pripojenia sa pridávali ďalšie volania servisov a dopĺňali sa jednotlivé procesy. V takýchto iteráciach a vždy po osobitnom unit teste nových servisov sa aplikácia budovala a priebežne testovala.
Zvoliť inú podobu integračného testu zatiaľ nie je v našich možnostiach, keďže finálna verzia EWS beží na platforme IBM WebSphere BPM. V rámci tejto platformy sú riešené základné problémy, ktoré by inak bolo nutné testovať.

5.3 Ekonomické a mimoekonomické zhodnotenie práce

EWS je určený pre špecifickú časť trhu - banky, pre ktoré má značnú hodnotu. Systém má potenciál výrazne zlepšiť kvantifikáciu a kvalifikáciu rizika, z ktorej vo všeobecnosti vyplývajú najväčšie zisky pre banku. Z tohto faktu vychádza jednoznačný záver, že produkt by bol zaujímavý pre klientov, za predpokladu, že by bol ďalej upravovaný a dopĺňaný, vrátane testovania v praxi.
Mnohé banky už využívajú rôzne systémy na hodnotenie klientov, riešenie nesplácania a podobne. EWS navrhnutý našim tímom by ponúkal jednotné užívateľské rozhranie, efektivitu a možnosť flexibilných úprav a zmien.
Riešenie je postavené na produktoch z rodiny IBM WebSphere. Tie však nie sú jedinou možnosťou a systém by bolo možné vybudovať aj na iných platformách. Rozoberme si jednotlivé súčasti EWS a ich alternatívy:

- **Databáza a dátový sklad** - okrem DB2 by sa na klasický relačný model dala použiť akákoľvek dostatočne rýchla iná databáza, či už komerčná, alebo voľne dostupná. Rozdiel by bol samozrejme v cene, rýchlosti a celkovej efektivite (napríklad pamäťová náročnosť či stabilita).

- **Enterprise Service Bus** - ESB podsystém by nemusel byť reprezentovaný iba IBM WebSphere Message Broker platformou, ale napríklad aj Oracle Service Bus alebo vlastným projektom. Pri vlastných riešeniach by došlo k úspore peňazí za samotný produkt, ale časový sklz, spôsobený návrhom, implementáciou a testom by bol značný.

- **Business Process Management** - IBM WebSphere BPM nie je jediný produkt, ktorý umožňuje správu procesov. Medzi alternatívy patrí napríklad komerčný Oracle Business Process Management či open-source nástroje (medzi ktoré patria Process Maker, Cuteflow a iné).

- **Obchodné pravidlá** - na obchodné pravidlá sa dajú využiť BRMS systémy, za alternatívu by sa dal považovať napríklad JBoss či DROOLS, prípadne nemusia byť obchodné pravidlá osobitne vyčlenené, ale súčasťou systému.

Podstatnou je skutočnosť, aby mali produkty, v ktorých má byť systém vytvorený, dostatočnú podporu zo strany ich tvorcov. Teoretické výkonnostné testy nie sú vždy jednoznačným a určujúcim ukazovateľom, dôležité je aj funkcionalita a možnosť prispôsobenia platforiem, ďalej fakt, či sú stále vo vývoji a iné.
Nástroj, ktorý som počas tvorby mojej diplomovej práce podrobne preskúmal, IBM WebSphere Message Broker, má veľkú devízu aj v integrácií systémov, ktoré sú nekonzistentné. Bolo by možné vykonať konverziu rôznych formátov dát či komunikáciu medzi nezávislými systémami. To by samo o sebe uľahčilo tvorbu EWS v už fungujúcom prostredí banky a aj s využitím rôznych iných produktov na BPM či správu obchodných pravidiel.
Medzi nevýhody IBM WebSphere Message Broker patrí okrem relatívne vyššej ceny aj veľkosť systému a neúplná funkcionalita (aj v oblasti komunikácie s databázami, grafickými mapami, tvorbou servisov či testu). V prípade aktívneho vývoja a konsolidácie by sa táto platforma mohla stať veľmi zaujímavou voľbou pre integráciu.

Záver

V závere mojej diplomovej práce by som chcel skonštatovať, že úlohy zo zadania boli úspešne splnené, teoretické oblasti spracované a EWS vytvorený a v práci spracovaný v oblasti návrhu, implementácie, nasadenia a testovania.

Cennou skúsenosťou pre mňa bola orientácia v zložitej problémovej oblasti, práca v komplexnom nástroji, ktorý je používaný v praxi. Za dôležité považujem spomenúť, že celá práca bola vyvíjaná v tíme študentov. Okrem mnohých nových skúseností z prostredia bánk, úverov a servisne orientovanej architektúry tak bolo dôležité riešiť aj tímovú spoluprácu, zdieľanie projektov a prostriedkov a efektívnu komunikáciu.

Za seba hodnotím túto spoluprácu ako veľmi dobrú a skúsenosti z nej sa budem snažiť využiť aj v ďalšom zamestnaní či štúdiu.

V budúcnosti by mohol byť systém EWS značne rozšírený, najmä v oblasti využitia dátového skladu, ktorý zatiaľ nebolo na platforme možné prakticky vytvoriť. Pre potreby logickej vrstvy by bolo možné vytvoriť ďalšie servisy, rozšíriť najmä oblasť logovania, získavania dát z externých zdrojov a iné.

Prácu na diplomovej práci ako celok považujem za prínos pre mňa v oblasti osobnostnej i vedomostnej. Na záver by som rád uviedol jeden citát od Júliusa Slováka, v ktorom hovorí: „*Ak sú tvoje spomienky väčšie než tvoje sny, už si začal zomierať*.“

Literatúra

[1] Holley K., Arsanjani A.: *100 SOA Questions: Asked and Answered*, Prentice Hall, 2011. ISBN 978-0-137-08020-5.

[2] Herzum, P., Sims, O.: *Business Component Factory : A Comprehensive Overview of Component-Based Development for the Enterprise*, Wiley, 1999. ISBN 978-0471327608.

[3] Bleakley, D., Chow, J., Clapperton K., Dayananda H. S., Vivek G., Sung S.: *Connecting Your Business Using WebSphere Message Broker V7 as an ESB*, International Technical Support Organization, 2010.

[4] Davies S., Cowen L., Giddings C., Parker H.: *WebSphere Message Broker Basics*, International Technical Support Organization, 2005.

[5] Rose R.: *Why WebSphere Message Broker Is Better Than Oracle Service Bus*, SWG Competitive Project Office, 2009.

[6] Dickens M., Kittel A., Leuckie R., Scaggs T., Thomas R.: *Mule ESB and JBoss ESB Cannot Keep Up*.

[7] *Computerworld: Ucelený informační zdroj pro IT profesionály.* Zodp. red. Vít Pertjanoš. Roč. 24. Praha: IDG Czech Republic, a.s., 2013. ISSN 1210-9924.

[8] Matiaško K., Vajsová M., Zábovský M., Chochlík M.: *Databázové Systémy: Základy databázových systémov*, EDIS, 2008. ISBN 978-80-8070-820-7.

[9] Lacko, Ľ.: *Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle*, Computer Press, 2003. ISBN 80-7226-969-0.

[10] Fowler, M.: *Analysis Patterns, Reusable object models*, Addison-Wesley Longman, 1997. ISBN 0-201-89542-0.

[11] *Service Oriented Architecture (SOA)* internetový zdroj: http://msdn.microsoft.com/en-us/library/bb833022.aspx, 9.3.2013.

[12] *Three-Layered Services Application* internetový zdroj: http://msdn.microsoft.com/en-us/library/ff648105.aspx, 9.3.2013.

[13] *Reference Architecture: The best of best practices*, internetový zdroj: http://www.ibm.com/developerworks/rational/library/2774.html, 18.4.2013.

[14] *New to WebSphere: Get acquainted with IBM's application and integration software* internetový zdroj: http://www.ibm.com/developerworks/websphere/newto/, 30.3.2013.

Prílohy

Príloha č.1 - Príklad XSD schémy webového servisu

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://Loan"
        targetNamespace="http://Loan">
   <xsd:element xmlns:ibmSchExtn="http://www.ibm.com/schema/extensions"
       name="LoanOperation" ibmSchExtn:docRoot="true">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="LoanRequest" nillable="true" type="tns:LoanRequest" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element xmlns:ibmSchExtn="http://www.ibm.com/schema/extensions"
   name="LoanOperationResponse" ibmSchExtn:docRoot="true">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="LoanResponse" nillable="true"
            type="tns:LoanResponse" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="LoanRequest">
      <xsd:sequence>
         <xsd:element name="ClientID" maxOccurs="1" minOccurs="1" type="xsd:int" />
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="LoanResponse">
      <xsd:sequence>
         <xsd:element name="Lien" type="tns:LienType" />
         <xsd:element name="Date" type="xsd:date" />
         <xsd:element name="TotalAmount" type="xsd:double" />
         <xsd:element name="NumberOfPaymentsLeft" type="xsd:int" />
         <xsd:element name="InterestRate" type="xsd:double" />
         <xsd:element name="Type" type="xsd:int" />
         <xsd:element name="AmountPerPeriod" type="xsd:double" />
         <xsd:element name="FixationStartDate" type="xsd:date" />
         <xsd:element name="FixationEndDate" type="xsd:date" />
```

```
        <xsd:element name="PaymentPeriod" type="xsd:int" />
        <xsd:element name="PaidAmount" type="xsd:double" />
        <xsd:element name="DateOfLasRepayment" type="xsd:date" />
        <xsd:element name="NumbeOfPeriods" type="xsd:int" />
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="LienType">
      <xsd:sequence>
        <xsd:element name="Type" type="xsd:int" />
        <xsd:element name="Name" type="xsd:string" />
        <xsd:element name="Value" type="xsd:double" />
        <xsd:element name="ValidFrom" type="xsd:date" />
        <xsd:element name="ValidUntil" type="xsd:date" />
        <xsd:element name="Date" type="xsd:date" />
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

Príloha č.2 - Príklad WSDL definície webového servisu

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:tns="http://Loan" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        name="Loan" targetNamespace="http://Loan">
   <wsdl:documentation>
      <wsdl:appinfo source="WMQI_APPINFO">
        <MRWSDLAppInfo imported="true">
           <generatedXSD location="Loan_InlineSchema1.xsd" />
           <binding name="LoanHttpBinding" hasEncoding="false" imported="true"
            originalBindingStyle="document" />
        </MRWSDLAppInfo>
      </wsdl:appinfo>
   </wsdl:documentation>
   <wsdl:types>
      <xsd:schema targetNamespace="http://Loan">
        <xsd:include schemaLocation="http://192.168.21.30:7805/Loan/Loan?xsd=xsd0" />
      </xsd:schema>
   </wsdl:types>
   <wsdl:message name="LoanOperationRequestMsg">
      <wsdl:part name="LoanOperationParameters" element="tns:LoanOperation" />
   </wsdl:message>
   <wsdl:message name="LoanOperationResponseMsg">
      <wsdl:part name="LoanOperationResult" element="tns:LoanOperationResponse" />
   </wsdl:message>
   <wsdl:portType name="Loan">
      <wsdl:operation name="LoanOperation">
        <wsdl:input name="LoanOperationRequest"
              message="tns:LoanOperationRequestMsg" />
        <wsdl:output name="LoanOperationResponse"
              message="tns:LoanOperationResponseMsg" />
      </wsdl:operation>
```

```
   </wsdl:portType>
   <wsdl:binding name="LoanHttpBinding" type="tns:Loan">
      <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="LoanOperation">
         <soap:operation soapAction="http://Loan/LoanOperation" />
         <wsdl:input name="LoanOperationRequest">
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="LoanOperationResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="LoanHttpService">
      <wsdl:port name="LoanHttpPort" binding="tns:LoanHttpBinding">
         <soap:address location="http://192.168.21.30:7805/Loan/Loan" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```