INTEGRACIJA – KLJUČ DO UČINKOVITEGA INFORMACIJSKEGA SISTEMA

Matjaž B. Jurič, Ivan Rozman
Univerza v Mariboru, Inštitut za informatiko, Fakulteta za elektrotehniko, računalništvo in informatiko
E-pošta: matjaz.juric@uni-mb.si, URL: http://lisa.uni-mb.si/~juric/

Izvleček

Parcialni, segmentirani informacijski sistemi v obliki množice nepovezanih ali delno povezanih aplikacij danes ne morejo zadovoljiti naraščajočih potreb, ki jih povzročajo elektronsko poslovanje in globalna povezljivost na eni ter nova ekonomija na drugi strani. Integracija je ključen pristop, s pomočjo katerega v časovno sprejemljivem okviru povečamo učinkovitost informacijskega sistema podjetja, hkrati pa ohranimo obstoječe rešitve. V prispevku se bomo osredotočili na organizacijsko strateški in tehnološki vidik integracije. Prikazali bomo večfazni postopek integracije, ki temelji na vzorcu integracijskega posrednika in način ovijanja obstoječih sistemov.

Abstract

Partial, segmented information systems, constituted of several non-connected or partially connected applications cannot meet the increasing demands, initiated by electronic commerce and global connectivity on one hand and new economy on the other hand. Integration is the key process, which enables us to increase the enterprise information system efficiency in a reasonable time and allows us to preserve existing (legacy) solutions. In the article we focus on organizational, strategic and technological viewpoints of integration. We present a multiphase integration process, based on the integration broker pattern. We also present a technique for wrapping of existing systems.

■■■

1. Uvod

Sodobne informacijske sisteme zaznamujejo heterogena zasnova, komponentna zgradba, lokacijska neodvisnost in povezljivost. Samostojne, monolitne aplikacije postajajo vedno bolj redke, vedno bolj redki so tudi veliki informacijski sistemi, zgrajeni po specifičnih zahtevah naročnika. Sodobna podjetja kupujejo programsko opremo v obliki komponent, posamezne komponente pa nato integrirajo v celovit informacijski sistem [4, 6]. Ravno integracija delov informacijskega sistema v celoto je eden od najtrših orehov. Za uspešno izvedbo integracije, kot ključnega procesa za izkoriščanje prednosti sodobnih informacijskih sistemov, je potrebno določiti strategijo in izbrati ustrezno tehnološko bazo.

Pomen in vloga integracije delov informacijskega sistema v celoto je razvidna iz projekcije Gartner Group: »Do leta 2004 bo 70% družb, ki so ali bodo oblikovale centralizirano službo za integracijo aplikacij, izvedlo integracijo učinkovito. 70% družb, ki bo prepustilo odgovornost za integracijo posameznim aplikacijskim sektorjem, integracije ne bo uspešno izvedlo (verjetnost 0.8).«

Zaradi tega je integracijski arhitekturi potrebno posvetiti veliko pozornost. V tem prispevku obravnavamo organizacijsko-strateška in tehnološka vprašanja.

2. Organizacija

Iz izsledkov zgoraj navedene napovedi Gartner Group in sorodnih raziskav je razvidno, da sta za podjetje najpomembnejši vpeljava projekta integracije in organizacija centralne službe za integracijo. Takšna služba ima v splošnem dve odgovornosti. Prva je operativna in se navezuje na izbiro, instalacijo in vzdrževanje konsistentne infrastrukture za integracijo, ki je enaka za celotno podjetje. Ta infrastruktura zagotavlja visoko nivojsko integracijo, logično nad lokalnim omrežjem podjetja. Pri tem uporablja eno od vrst ali kombinacijo komunikacijskega vmesnega sloja (middleware) v obliki integracijskega posrednika [7].

Druga odgovornost se veže na razvoj in vzdrževanje dokumentacije za integracijo aplikacij, ki je po navadi sestavljena v obliki množice vmesnikov.

Potrebno je zagotoviti, da posamezne projektne skupine, ki razvijajo aplikacije, upoštevajo izbrano integracijsko paradigmo in v aplikacije vgradijo ustrezno podporo.

Zagotoviti je torej potrebno, da se integracija izvaja na nivoju celotnega podjetja in se vodi in koordinira centralno. Zato naj se vpelje projekt integracije in ustvari ustrezna služba, ki bo razvijala, vpeljevala in vzdrževala integracijsko infrastrukturo.

Tipičen informacijski sistem je sestavljen iz več modelov. Z vidika integracije je ključnega pomena model, ki opisuje relacije z zunanjim svetom. Večina starejših informacijskih sistemov je največjo pozornost posvečala modelom notranje organizacije. Relacije z zunanjim svetom pa so bile pogosto zanemarjene. Informacijski sistemi tudi niso skladni s celovitim objektnim in podatkovnim modelom podjetja. V praksi ni možno modificirati aplikacij tako, da bi zagotovili, da je njihova interna struktura konsistentna. Zato je za uspešno izvedbo integracije potrebno upoštevati omejitve in realno situacijo.

Za uspešno izvedbo integracije sama tehnologija ni zadostna. *Tehnoloških ovir za integracijo heterogenih sistemov danes praktično ni.* Po drugi strani pa imamo lahko velike težave pri integraciji dveh aplikacij na istem operacijskem sistemu, napisanih v istem programskem jeziku z isto podatkovno bazo. Integracija je torej odvisna od arhitekture aplikacije in od organizacije vmesnikov, prek katerih aplikacije komunicirajo z zunanjim svetom. Noben tehnološki standard ne more zaobiti organizacijskih problemov.

Uporaba informacijskih rešitev v obliki sistemov ERP (Enterprise Resource Planning) lahko izboljša, ne more pa ukiniti potreb po integraciji. To velja tudi za podjetja, ki kupijo vse od enega dobavitelja. Po ocenah raziskav, sistemi ERP običajno pokrijejo do 30 odstotkov poslovnih funkcij podjetja. Ostale naloge opravljajo obstoječi sistemi, lastne majhne aplikacije in programska oprema drugih dobaviteljev.

Ker je integracija sistemov ERP zahtevna, dobavitelji teh sistemov dopolnjujejo svoje rešitve z vpeljavo podpore za predvsem komponentne vmesnike. S tem se strinja tudi GartnerGroup, katerega prognoza pravi: »Od 1999 do 2001 bodo vsi glavni dobavitelji ERP ponudili bistveno boljše vmesnike do posrednikov zahtev objektov in sporočilnih sistemov, kar bo omogočilo integracijo z zunanjimi aplikacijami (verjetnost 0.8).«

Ključne aplikacije običajno izkazujejo velike arhitekturne in implementacijske razlike, ki so razvidne v različnih uporabniških vmesnikih, načinu komunikacije, strukturi podatkov in dostopa do podatkovne baze. Pomanjkljivosti glede integracije so:

- heterogenost in neusklajenost uporabniških vmesnikov,
- nepovezanost delov informacijskega sistema,
- prekrivanje dela funkcionalnosti,
- redundantnost podatkov,
- klasična arhitektura odjemalec-strežnik ali celo monolitna zgradba,
- neobstoj standardnih, jasno definiranih vmesnikov za povezljivost.

V pogojih globalizacije tržišča, globalne povezljivosti in dostopnosti informacij in prehoda na elektronsko poslovanje bodo informacijski sistemi morali zagotoviti nove oblike uporabe in se prilagoditi novim zahtevam. Arhitekture, tehnologije in načini vzdrževanja, ki so dobro delovali za klasične, ročno izdelane in nepovezane aplikacije, se izkažejo za neučinkovite in celo kontraproduktivne, če jih apliciramo na današnje heterogene in porazdeljene sisteme.

Le če bo podjetje sposobno hitro prilagoditi obstoječe aplikacije in uporabiti njihovo funkcionalnost na nove načine, bo uspešno delovalo v nastajajoči dobi elektronskega poslovanja. Uspešnost bo odvisna od uporabe novih vzorcev načrtovanja in načinov upravljanja pri zagotavljanju integracije. Izvajanje poslovnih strategij, kot sta »podjetje brez zakasnitve« (Zero Latency Enterprise) ali »premočrtno procesiranje« (Straight Through Processing), lahko zagotovijo le sodobni informacijski sistemi. Strategija »premočrtno procesiranje« zagotavlja, da se isti podatki v informacijski sistem vnašajo samo enkrat. S tem je zagotovljena konsistentnost podatkov in minimirana njihova redundanca. Strategija »podjetje brez zakasnitve« zagotavlja, da se spremembe, narejene v delu informacijskega sistema, takoj (sinhrono), torej brez zakasnitve, odrazijo v ostalih delih celotnega informacijskega sistema, ne glede na njegovo distribuiranost. Običajno ločimo sisteme, ki implementirajo enostransko obveščanje o dogodkih, in sisteme, ki implementirajo dvosmerni kompozitni integracijski vzorec. Tradicionalni občasni paketni prenosi podatkov med aplikacijskimi sistemi in podatkovni bazami ne zadoščajo.

3. Tehnologija

V produkcijskih informacijskih sistemih najdemo danes v številnih primerih paketno komunikacijo z izmenjavo datotek. Taka komunikacija je nedvomno naprednejša od ročnega vnosa podatkov in posledično nesinhroniziranih podatkovnih baz. Kljub temu pa tak način vedno težje vzdrži pritisku naraščajočih poslovnih potreb, ki zahtevajo vedno bolj ažurne podatke [1]. Tako se dogaja, da se paketna komunikacija zaganja večkrat na uro, da bi se zagotovila konsistenca. Poleg tega je direkten prenos v podatkovne baze problematičen, saj zaobide programsko

logiko in poslovna pravila informacijskih sistemov, ki podatke sprejemajo, kar lahko vodi do nekonsistentnosti v podatkih.

Zaradi tega je za integracijo heterogenih informacijskih sistemov uveljavljen *vzorec integracijskega posrednika*. Tak posrednik je osrednji komunikacijski vmesni sloj med aplikacijskimi sistemi ne glede na njihovo tehnološko osnovo. Uporabi omenjenega vzorca govori v prid tudi projekcija Gartner Group: »Do 2001 bo imela več kot polovica velikih podjetij vpeljano eno od oblik integracijskega posrednika (verjetnost 0.8).«

Integracijski posrednik prinaša številne prednosti:

- nudi vnaprej izdelano komunikacijsko infrastrukturo, ki je neodvisna od tehnološke osnove,
- nudi preverjeno in učinkovito podlago za integracijo,
- loči direktno komunikacijo med posameznimi aplikacijami in kardinalnost N-proti-N zmanjša na 1-proti-N,
- omogoča izvedbo integracije z upravljanjem povezav med sistemi na način črne škatle.

Integracijski posredniki delimo na dve skupini, ki se razlikujeta po načinu delovanja in povezave:

- posredniki zahtev objektov (ORB – Object Request Broker) in
- sporočilni sistemi (MOM – Message Oriented Middleware).

Posredniki zahtev objektov so infrastruktura za komunikacijo med porazdeljenimi objekti oziroma komponentami [3]. Zasnovani so kot transportni sistem, na katerem temeljijo sodobni pristopi k gradnji informacijskih sistemov. Posredniki zahtev objektov so osnova za večslojno arhitekturo aplikacij in temelj komponentnih modelov. Ponujajo različne oblike komunikacije – tako sinhrono kot tudi asinhrono komunikacijo med enotami sistema [5]. Zaradi pomena posrednikov zahtev objektov, ki znatno presegajo le okvire integracije aplikacij, smo bili v zadnjih letih priča standardizaciji in danes obstajata dva pomembna standarda:

- Družina posrednikov zahtev objektov arhitekture CORBA (Common Object Request Broker Architecture), ki jo je standardiziral OMG (Object Management Group)in deluje preko protokola IIOP (Internet Inter-ORB Protocol). V to družino prištevamo poleg številnih implementacij arhitekture CORBA tudi porazdeljen objektni model platforme Java 2: RMI oz. RMI-IIOP (Remote Method Invocation).
- Posrednik zahtev objektov Microsoftove arhitekture DCOM/COM+, ki deluje na osnovi protokola ORPC, razširjenega protokola RPC modela OSF DCE (Distributed Computing Environment).

Posredniki zahtev objektov (predvsem družine CORBA) so standardizirani [13, 14]. To je posebej pomembno, saj se z njihovo uporabo odločimo za splošno sprejet industrijski standard in ne za rešitev posameznega podjetja. Poleg tega so omenjeni posredniki zahtev objektov tudi osnova, na kateri delujejo komponentni modeli druge generacije (CCM, EJB in COM+) [4, 8].

Sporočilni sistemi ponujajo le asinhroni način komunikacije med enotami sistema. Delujejo na osnovi sporočilnega kanala. Njihov pomen slabi, saj njihovo funkcionalnost praktično v celoti nudijo tudi posredniki zahtev objektov in v prihodnosti pričakujemo zlitje obeh tehnologij. Prav tako na področju sporočilnih sistemov nismo bili priča standardizaciji. Posledica je, da sistemi različnih proizvajalcev med seboj niso združljivi z vidika programske kode. Pri spremembi sporočilnega sistema moramo spreminjati tudi naše aplikacije. Edini resnejši poskus abstrakcije je bil narejen v Java 2 Enterprise Edition, kjer je definiran univerzalen vmesnik JMS (Java Messaging Service). Omenimo le dva pomembna predstavnika sporočilnih sistemov: IBM MQSeries in MSMQ.

Pomembna razlika med posredniki zahtev objektov in sporočilnimi sistemi je tudi v zmogljivostih. Odzivni časi posrednikov zahtev objektov so praviloma krajši, že zaradi načina komunikacije [10, 11, 12]. Zaradi že omenjenih potreb po podjetju brez zakasnitve in premočrtnemu procesiranju, ki nedvomno favorizirata sinhrono integracijo distribuiranih aplikacij, zaradi omenjenih boljših zmogljivosti ter dejstva, da so posredniki zahtev objektov ključna tehnologija sodobnih pristopov k gradnji informacijskih sistemov, se bomo v nadaljevanju prispevka osredotočili na posrednika zahtev objektov in porazdeljene objektne oziroma komponentne modele [15, 2].

4. Izbira posrednika zahtev objektov

Posredniki zahtev objektov so vrsta programske opreme za vmesni sloj (middleware). Omogočajo transparentno komunikacijo delov informacijskega sistema. Pri tem razvijalcem dopuščajo uporabo že poznane sintakse za komunikacijo med programskimi entitetami. Posredniki zahtev objektov skrijejo vse podrobnosti komunikacije in jo naredijo neodvisno od lokacije, programskih jezikov in orodij, operacijskih sistemov, platform ali omrežnih vmesnikov. Praviloma so sestavni del porazdeljenih objektnih oziroma komponentnih modelov.

Porazdeljeni objektni modeli omogočajo izgradnjo programske opreme z integracijo novo razvitih in ograjenih obstoječih komponent - objektov. Zagotavljajo enotno komunikacijsko infrastrukturo. Z zmožnostjo ponovne uporabe, prenosljivosti in povezljivosti v

distribuiranem, heterogenem omrežju rešujejo večino težav, s katerimi se danes srečujejo razvijalci programske opreme in uporabniki računalnikov. Poleg tega ponujajo množico storitev, kot npr. transakcijske, varnostne in relacijske storitve, storitve življenjskega cikla, trajnega stanja, poimenovanja in druge. Nudijo tudi podporo zagotavljanju razširljivosti.

Omenili smo že, da posrednike zahtev objektov delimo v družino CORBA (ki vključuje tudi Java RMI-IIOP) in Microsoft DCOM/COM+. Obe družini temeljita na podobnih osnovnih konceptih [6, 9], razlike pa obstajajo v številnih podrobnostih. Glede integracijskega posrednika je za omenjene modele najpomembnejše dejstvo podpore različnim operacijskim sistemom in programskim jezikom.

Tabela 1 prikazuje podporo operacijskim sistemom za omenjene porazdeljene objektne modele. Ker implementacije modela CORBA ponujajo različni dobavitelji, so podprti vsi relevantni operacijski sistemi. Povezljivost med različnimi implementacijami modela CORBA in z modelom RMI-IIOP je zagotovljena s podporo protokolu GIOP/IIOP (General Inter-ORB Protocol/Internet Inter-ORB Protocol).

CORBA	Windows (vse verzije, tudi 2000), Mac, Solaris, SunOS, SGI, HPUX, OS/2, SCO, Irix, Digital Unix, AIX, VMS, VxWorks, QNX, MVS, OS/400, pSOS, Cray, Linux[1]
Java RMI-IIOP	Vsi s podporo za JVM (Java Virtual Machine)
DCOM/COM+	Windows NT 4, 2000, 95, 98

Tabela 1: Podpora za operacijske sisteme

Tabela 2 prikazuje podporo za programske jezike.

CORBA C#[2]	C, C++, Java, COBOL, Smalltalk, Ada, PLI,
RMI-IIOP	Java[3]
DCOM/COM+	C, C++, C#, Java, Visual Basic; ostali jeziki preko Automation

Tabela 2: Podpora za programske jezike

Poleg omenjenih je pri izbiri potrebno posvetiti pozornost vsaj naslednjim kriterijem. Zrelost modela, ki se izraža v prisotnosti na tržišču, omogoča sklepanje o njegovi zanesljivosti. Jezikovno neodvisen protokol je pomemben za izdelavo komunikacijske hrbtenice. Zelo pomembna kriterija sta enostavnost učenja in enostavnost razvoja programske opreme in ju velikokrat ni enostavno vnaprej oceniti.

Poleg tega je pri izbiri potrebno pozornost posvetiti tudi temu, kako se arhitektura sklada z deli informacijskega sistema, ki smo ga razvili v hiši in kako arhitekturo podpirajo dobavitelji ostalih delov sistema. Predvsem je to pomembno pri velikih sistemih, kot so sistemi ERP. SAP tako že nekaj časa podpira tako CORBA, kakor tudi DCOM/COM+. Zanemariti ne smemo tudi obstoječe baze znanja zaposlenih.

V nadaljevanju si bomo ogledali načine integracije obstoječih sistemov v enovit informacijski sistem in postopek ovijanja delov informacijskega sistema.

5. Večfazni postopek integracije

Ugotovili smo že, da je v podjetjih običajno del programske opreme kupljen na tržišču, del pa samostojno razvit (ali znotraj podjetja ali skladno s specifikacijami podjetja pri zunanjih razvijalcih). Poleg tega obstajajo lokalne in ad-hoc rešitve. Ključne aplikacije običajno izkazujejo velike arhitekturne in implementacijske razlike, ki so razvidne v različnih uporabniških vmesnikih, načinu komunikacije, strukturi podatkov in v dostopu do podatkovne baze. Prav tako smo ugotovili, da je za integracijo smiselna uporaba integracijskega posrednika.

Predlagani postopek integracije informacijskih sistemov je inkrementalen. Vsebuje tri glavne faze, poimenovane kot:

- konsistentnost podatkov,
- večnivojski proces in
- kompozitni informacijski sistem.

5.1. Konsistentnost podatkov

Cilj prve faze integracije je doseči konsistentnost podatkov na logičnem nivoju. Podatki so redundantni, integracija pa zagotavlja njihovo prenašanje med posameznimi aplikacijami. Aplikacije so neodvisne, povezave se ne zavedajo in imajo ločeno in neodvisno definirane procese. Takšno strukturo prikazuje slika 1.

Značilnosti:

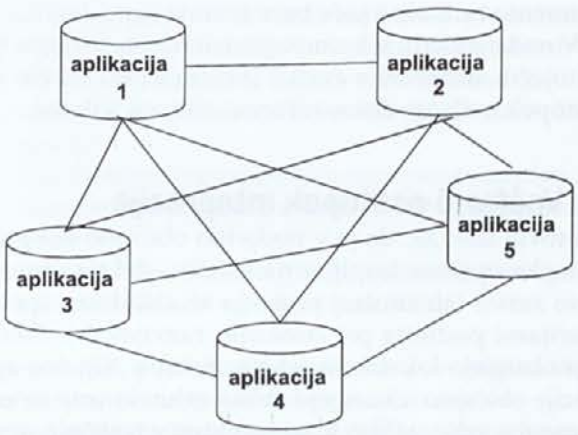
- ločeni poslovni procesi,
- potrebnih je več korakov prenosa,
- komunikacija je enosmerna in asinhrona,
- prenosi so običajno paketni,
- sistemi so fizično neodvisni,
- sistemi so logično povezani, vendar se tega ne zavedajo,
- delna skladnost s strategijo premočrtnega procesiranja.

1 *Našteti so samo najpomembnejši operacijski sistemi. Obstaja tudi podpora za druge, manj pomembne operacijske sisteme.*

2 *Našteti so le jeziki, z katere je podpora standardizirana s strani OMG. Podprti so tudi drugi jeziki, kot npr. Objective C, Perl, Delphi, Visual Basic, Eiffel, Common Lisp, Scheme [13].*

3 *Java vsebuje omejeno podporo za C++ preko vmesnika JNI (Java Native Interface).*

Integracija za dosego konsistentnosti podatkov se implementira s prenosom podatkov med aplikacijami. Pri tem je potrebno poznati podatkovne modele aplikacij in identificirati podatke, ki so skupni dvema ali več aplikacijam. Prek ustreznih API-jev ali protokolov je potrebno dostopiti do izvorne podatkovne baze in ustrezne podatke prenesti v ciljno bazo. Pri tem je potrebno posebno pozornost posvetiti konsistenci podatkov in njihovi skladnosti s poslovnimi pravili, saj poslovna logika aplikacije ne nadzoruje direktnega dostopa do podatkovne baze .



Slika 1: Konsistentnost podatkov na logičnem nivoju

Ključni problem pri takšni integraciji je identifikacija ustreznih podatkov. Slabosti direktnega prenosa med podatkovnimi bazami so predvsem v načinu povezave vsak z vsakim, kar pri večjem številu sistemov privede do kombinatorične eksplozije. Osnovni problem je vzdrževanje in nadziranje velikega števila povezav. Dodatne slabosti so potreba po intervenciji (človeški ali avtomatizirani), paketnemu prenosu in posledičnim zakasnitvam in neažurnosti podatkov. Problem je tudi možnost, da uporabniki modificirajo podatke v samo eni aplikaciji.

Del omenjenih težav rešuje izvedba integracija na osnovi konsistence podatkov z uporabo integracijskega posrednika, v našem primeru posrednika zahtev objektov. Shemo take integracije prikazuje slika 2. Osnovna prednost je definicija vmesnikov do posameznih aplikacijskih sistemov in ločitev neposredne komunikacije med aplikacijami. Način definicije vmesnikov in ovijanja aplikacij si bomo ogledali v nadaljevanju.

S tem se kardinalnost iz N-proti-N zmanjša na 1-proti-N. Komunikacija preko posrednika zahtev objektov je lahko asinhrona, sinhrona ali odložena sinhrona. Če je dogodkovno orientirana, ponuja tudi korak k zagotavljanju strategije podjetja brez zakasnitev.

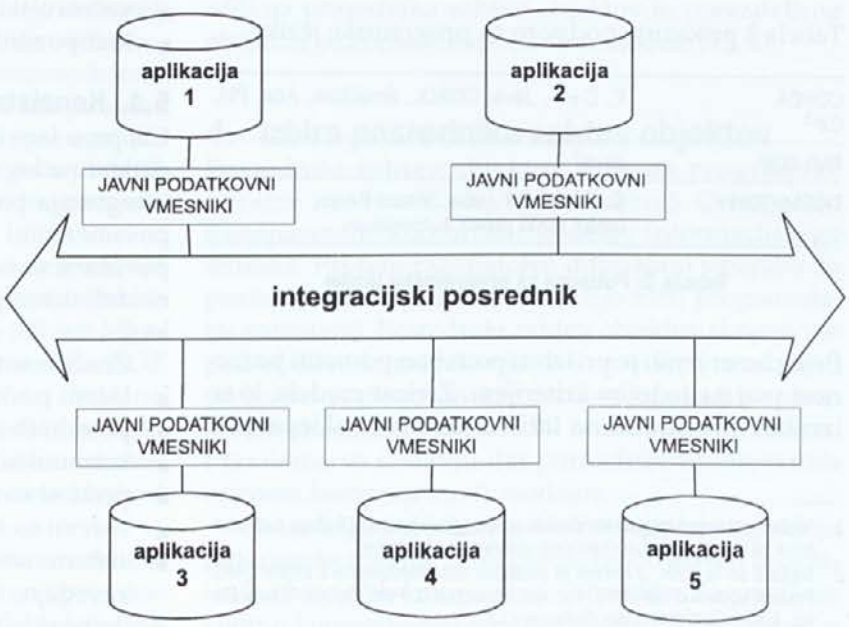
V drugem koraku prve faze izvedbe integracije tako predlagamo, da se za dele informacijskega sistema definirajo javni vmesniki, preko katerih bo možna sinhronizacija podatkov. Sam prenos se naj izvede s pomočjo posrednika zahtev podatkov. Ta način pred direktnim prenosom podatkov med aplikacijami nudi naslednje prednosti:

- oblikovati se začne infrastruktura za globalno integracijo aplikacij,
- začne se z definicijo javnih vmesnikov do podatkovne in poslovne logike aplikacij,
- olajša se nadzor in vzdrževanje komunikacije,
- komunikacija lahko temelji na dogodkih, ki jih prožijo aplikacije.

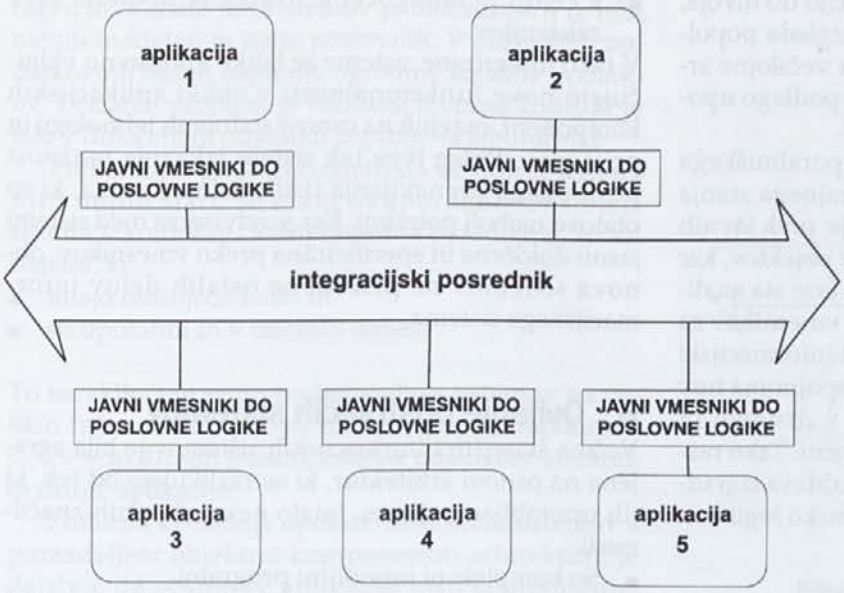
Razume se, da proces integracije ne zahteva popolne implementacije prvega koraka, ampak se lahko direktno pristopi k implementaciji arhitekture, predstavljene v drugem koraku.

5.2. Večnivojski proces

Druga faza integracije informacijskih sistemov se imenuje večnivojski proces, ki je prikazana na sliki 3. Večnivojski proces zahteva izdelavo globalnega modela načrtovanja informacijskega sistema. Tak model temelji na proučitvi scenarijev uporabe obstoječih aplikacij informacijskega sistema, na izdelavi primerov uporabe in na definiciji modela analize. Model



Slika 2: Konsistentnost podatkov z uporabo integracijskega posrednika



Slika 3: Večnivojski proces integracije

analize in analiza obstoječih aplikacij sta predpogoj za izdelavo modela načrtovanja ali globalnega objektnega modela informacijskega sistema. Pri tem je pomembno zagotoviti, da se upošteva vsa funkcionalnost obstoječih sistemov, hkrati pa se predvidijo manjkajoče funkcionalnosti, dopolnitve in spremembe. Globalni objektni model je najpomembnejše izhodišče za uspešno izvedbo druge faze integracije.

Slika 4: Oris aktivnosti za drugo fazo integracije

Na osnovi globalnega objektnega modela se definirajo javni vmesniki do obstoječih in do novo razvitih aplikacij. Tako definirani vmesniki nam skupaj z izbiro in uporabo ustrezne tehnologije omogočajo izdelavo objektnih ovojev okrog obstoječih aplikacij. S tem omogočimo dostop do poslovne logike obstoječih in novih aplikacij na enoten, univerzalen in transparenten način, ki je neodvisen od lokacije, operacijskega sistema, programskega jezika in strojne osnove. Diagram aktivnosti prikazuje slika 4.

Pri tem je potrebno poudariti, da je istočasno potrebno v proces razvoja programske opreme znotraj podjetja vnesti ustrezne spremembe in zagotoviti, da bo lastno razvita programska oprema upoštevala načela integracije in bo skladna z globalnim objektnim modelom. Zaradi tega predlagamo, da se za lastne projekte čim prej predvidijo načini in možnosti integracije, da se definirajo in implementirajo ustrezni vmesniki in se predvidi načrt podpore ustrezne integracijske tehnologije, v tem primeru posrednikov zahtev objektov. Za obstoječe kupljene komercialne sisteme je potrebno najti načine, kako s čim manj stroški čim bolj učinkovito zagotoviti vmesnike do poslovne logike.

Integracija na osnovi večnivojskega procesa zagotavlja:

- enoten poslovni proces,
- komunikacija je obojestranska, vendar običajno asinhrona,
- prenosi se lahko izvajajo v trenutku (ali po potrebi paketno),
- sistemi so fizično povezani,
- sistemi so logično povezani,
- delno je omogočena strategija premočrtnega procesiranja,
- delno je omogočena strategija podjetja brez zakasnitev.

Slabosti:

- neenoten uporabniški vmesnik,
- še vedno je potrebnih več korakov za realizacijo integracije,
- komunikacija še vedno poteka po paradigmi odjemalec-strežnik.

5.3. Kompozitni informacijski sistem

Kljub številnim prednostim druga faza integracije ni zagotovila enotnega in celovitega informacijskega sistema. Šele tretja faza integracije, imenovana kompozitni

informacijski sistem, zagotavlja integracijo do nivoja, ko informacijski sistem za uporabnika zgleda popolnoma enoten. Ta faza temelji na vzorcu večslojne arhitekture informacijskih sistemov in za podlago uporablja rezultate prejšnjih faz integracije.

Osnovni princip je stroga ločitev uporabniškega vmesnika, poslovne logike in nivoja trajnega stanja podatkov ter konsistentne komunikacije prek javnih vmesnikov na osnovi posrednika zahtev objektov, kar prikazuje slika 5. Osnovni dejavnosti te faze sta analiza in načrtovanje enotnih uporabniških vmesnikov za celovit informacijski sistem. Tako razvit informacijski sistem bo za uporabnike izgledal kot popolnoma nov sistem. V resnici pa bodo v ozadju, v drugem in tretjem sloju, delovali oviti obstoječi sistemi. Tako razvit kompozitni informacijski sistem ne zahteva izgradnje iz nič, ampak uporablja vso programsko logiko in obstoječe znanje, hkrati pa zagotavlja:

- enoten poslovni proces,
- popolno integracijo aplikacij, ki navzven delujejo kot ena velika aplikacija,
- enoten uporabniški vmesnik,
- dvosmerno, sinhrono komunikacijo,
- neposredne, takojšnje in individualne prenose podatkov,
- fizično odvisnost med sistemi,
- logično odvisnost med sistemi,
- v celoti je omogočena strategija premočrtnega procesiranja,
- v celoti je omogočena strategija podjetja brez zakasnitev.

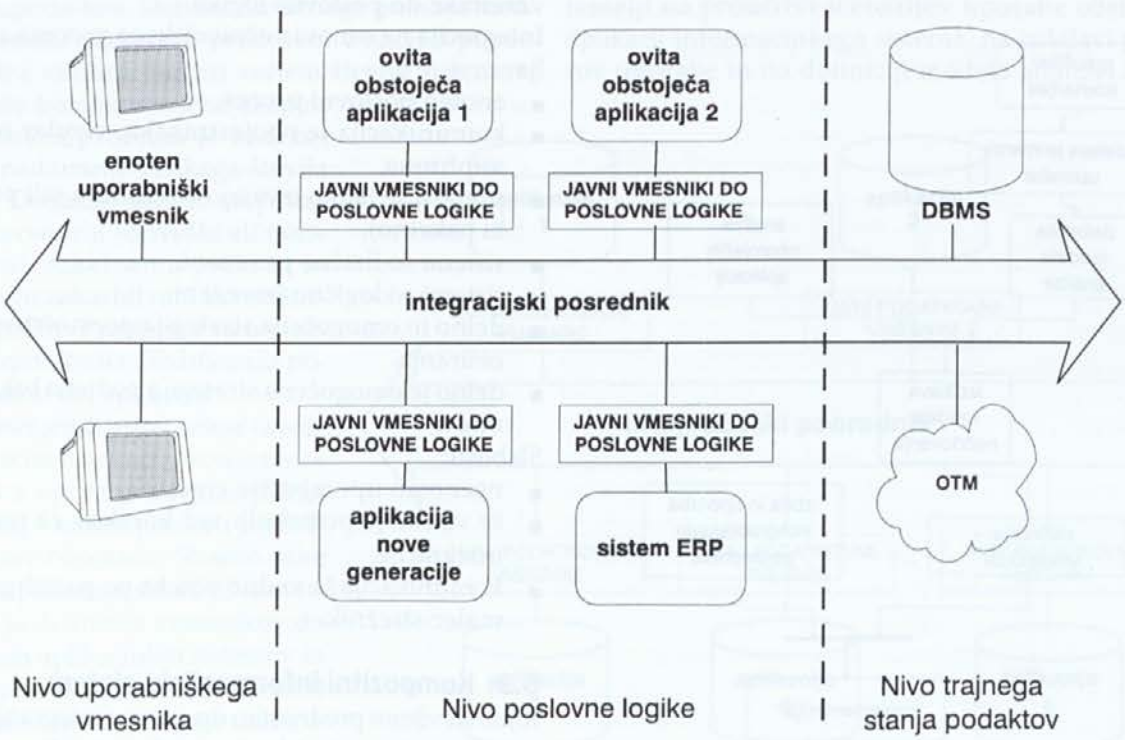
V tako integrirane sisteme se lahko enostavno vključujejo nove funkcionalnosti v obliki aplikacijskih komponent, razvitih na osnovi sodobnih tehnologij in pristopov. Poleg tega tak sistem izkazuje možnost postopnega prenavljanja tistih delov sistema, ki so obnove najbolj potrebni. Ker je odvisnost med sistemi jasno določena in specificirana preko vmesnikov, obnova sistemov ne prizadene ostalih delov informacijskega sistema.

6. Ovijanje obstoječih sistemov

Večina starejših informacijskih sistemov je bila zgrajena na osnovi arhitektur, ki se razlikujejo od teh, ki jih uporabljamo danes. Imajo nekaj skupnih značilnosti:

- So kompleksni monolitni programi.
- So ključni za opravljanje določenih nalog.
- Odvisni so od področja.
- Uporabljajo tradicionalne podatkovne baze ali pa še tega ne.
- Ponavadi delujejo na srednjih in velikih računalnikih in imajo centralizirano osebje, ki skrbi za razvoj, vzdrževanje in podporo.

Obstoječi sistemi so bili razviti s ciljem služiti posameznim poslovnim področjem, ne pa celemu podjetju. Tako imajo omejen doseg in se niso zmožni



Slika 5: Kompozitni informacijski sistem

razvijati. Večina teh sistemov je bila razvitih iz tehničnih modelov in ne iz poslovnih. V obstoječih podatkovnih bazah najdemo ogromne količine podatkov. Dodaten problem je neobstoj izvorne kode, ki je bila v določenem odstotku izgubljena ali uničena.

Pomembno pa je razumeti, da obstoječi sistemi niso nujno stari, še manj neuporabni. Obstoječi sistemi v kontekstu komponentne paradigme, so vsi sistemi, ki:

- imajo obstoječo kodo in
- so uporabni in v uporabi danes.

To ne vključuje samo tradicionalnih sistemov na velikih računalnikih pač pa tudi programe, napisane v C, C++ in drugih jezikih, sisteme odjemalec-strežnik in druge aplikacije.

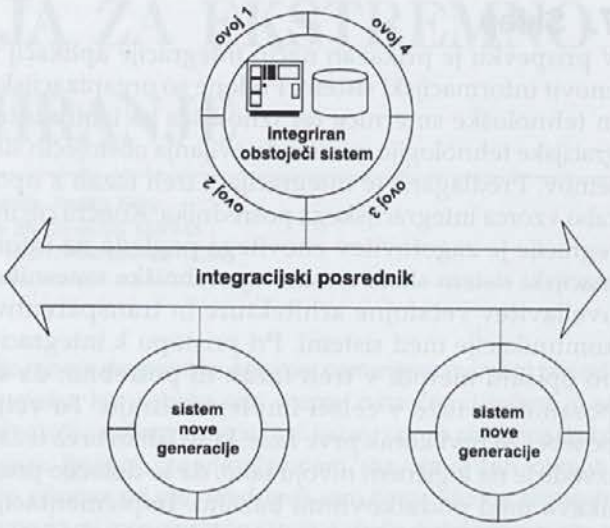
Problem, ki otežuje uporabo obstoječih sistemov v porazdeljeni objektno-komponentni arhitekturi, je dejstvo, da obstoječe aplikacije niso komponente. Izziv je najti način, kako ograditi obstoječe aplikacije tako, da bodo izgledale kot porazdeljene komponente in se bodo operacije ene komponente lahko implementirale v več aplikacijah.

Cilj integracije obstoječih sistemov je torej uporaba njihovih informacij in kode pri razvoju arhitekture informacijske podpore za podjetje. Želimo si, da bi obstoječi sistemi lahko komunicirali z drugimi obstoječimi in novo razvitimi sistemi (slika 6). Potrebujemo torej informacijsko arhitekturo, ki bi omogočila sodelovanje obstoječih sistemov in jih razbila v komponente, poslovne procese pa v ločena, kooperativna področja.

Takšno ločevanje v področja zahteva razumevanje vsebine sistemov in primerov njihove uporabe in implementacijo abstraktnih vmesnikov, ki omogočajo drugim področjem dostop do vsebine in primerov uporabe.

Integracija skrije obstoječe sisteme za konsistentnimi vmesniki, ki skrivajo implementacijske podrobnosti, so v skladu s poslovnim procesom in slovarjem in omogočajo spreminjanje ali zamenjavo implementacij brez vpliva na preostanek sistema. Abstraktni vmesnik ograjuje in skriva dejansko implementacijo sistema. Odjemalci dostopajo v sistem samo skozi ta vmesnik.

Ovijanje *(wrapping)* obstoječega sistema je proces, v katerem definiramo in omogočimo dostop do obstoječega sistema skozi abstraktni vmesnik. Na eni strani ovoj predstavlja sistem z abstraktnim vmesnikom. Na drugi strani ovoj komunicira z obstoječim sistemom,

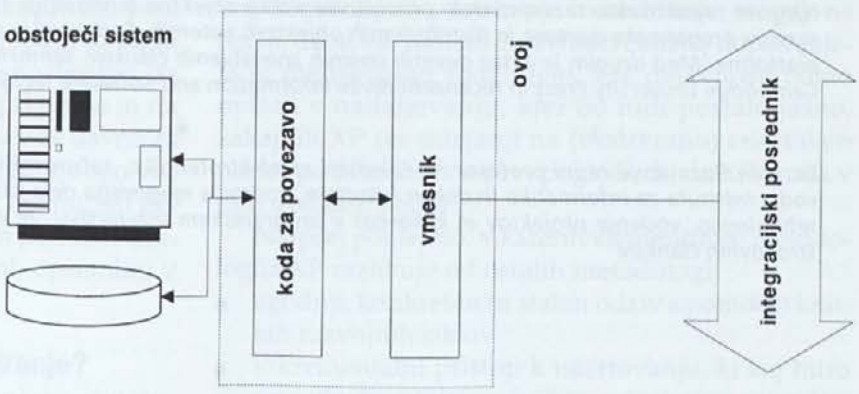


Slika 6: Integracija obstoječih sistemov

kot prikazuje slika 7. Ovoji predstavljajo sistemski pogled, neodvisen od implementacije. So naraven način za integracijo obstoječih sistemov drugega z drugim in z novimi sistemi. Enkrat ovit, se lahko obstoječi sistem vključi v porazdeljeno objektno-komponentno okolje z uporabo posrednikov zahtev objektov.

Na Univerzi v Mariboru, Inštitutu za informatiko, Centru za objektno tehnologijo imamo izkušnje z integracijo obstoječih sistemov. Definirali smo tehniko, ki natančno določa potrebne korake pri izdelavi ovoja in opozarja na področja, ki jim je potrebno posvetiti dodatno pozornost. Tehniko smo uspešno preizkusili pri ovijanju več realnih aplikacij [5, 7].

Čez čas lahko funkcionalnost obstoječih aplikacij postopoma zamenjamo s komponentami, dokler prvotna koda ne postane odvečna. Izbrane dele lahko obnavljamo korak za korakom, glede na potrebe in razpoložljiva sredstva. Tak pristop zmanjša stroške, poveča zmogljivosti in olajša vzdrževanje.



Slika 7: Ovijanje obstoječega sistema

7. Sklep

V prispevku je prikazan način integracije aplikacij v enovit informacijski sistem. Podane so organizacijske in tehnološke smernice ter izhodišča za izbiro integracijske tehnologije in metode ovijanja obstoječih sistemov. Predlagana je integracija v treh fazah z uporabo vzorca integracijskega posrednika. Končni cilj integracije je zagotovitev enovitega pogleda na informacijski sistem skozi enotne uporabniške vmesnike, uveljavitev večslojne arhitekture in transparentne komunikacije med sistemi. Pri pristopu k integraciji po opisani metodi v treh fazah ni potrebno, da se posamezne faze v celoti implementirajo. To velja posebej za prvi korak prve faze, ki se lahko brez težav izvede le na logičnem nivoju tako, da se določijo preslikave med podatkovnimi bazami. Implementacija prvega koraka ni potrebna in se lahko takoj pristopi k drugemu koraku prve faze. Končni cilj je izdelati arhitekturo, skladno s shemo, opisano v tretji fazi. Tako integriran informacijski sistem zagotavlja premočrtno procesiranje in distribucijo podatkov brez zakasnitev. Na ta način bo informacijski sistem prilagojen vstopu v globalno elektronsko ekonomijo informacijske družbe.

Viri

[1] Donald E. Rimel Jr., Ronald C. Aronica, *Leveraging Legacy Assets*, First Class, Dec 1996, OMG

[2] Juric B. Matjaz, Zivkovic Ales, Rozman Ivan, *Are Distributed Objects Fast Enough*, More Java Gems, Cambridge University Press, 2000

[3] Jurič M. B., Heričko M., Rozman I., *CORBA - objektni model porazdeljenega procesiranja*, Uporabna informatika, jan/feb/mar 1997

[4] Jurič B. Matjaž, *Nova generacija komponentnih modelov CORBA 3, COM+, EJB*, Objektna tehnologija v Sloveniji OTS'2000, 2000, str. 18-29

[5] Jurič B. Matjaž, Rozman Ivan, Heričko Marjan, *A method for integrating legacy systems within distributed object architecture*, International Conference on Enterprise Information Systems, Setúbal, Portugal, March 1999

[6] Jurič B. Matjaž, Rozman Ivan, Heričko Marjan, *CORBA komponente in pomen objektnih transakcijskih storitev*, Dnevi slovenske informatike, DSI 2000, del. 1, str. 52-60

[7] Jurič B. Matjaž, Rozman Ivan, Heričko Marjan, *Integrating legacy systems in distributed object architecture*, ACM Software Engineering Notes, March 2000, vol. 25, no. 2, str. 35-39

[8] Jurič B. Matjaž, Rozman Ivan, Heričko Marjan, *Performance comparison of CORBA and RMI*, Information and Software Technology Journal, 2000, vol. 42, no. 13, str. 915-933

[9] Jurič B. Matjaž, Rozman Ivan, *Java 2 RMI and IDL comparison*, Java Report, February 2000, vol. 5, no. 2, str. 36-48

[10] Jurič B. Matjaž, Rozman Ivan, Stevens Alan, Heričko Marjan, Nash Simon, *Java 2 distributed object models performance analysis, comparison and optimization*, International Conference on Parallel and Distributed Systems, California, IEEE Computer Society Press, 2000, str. 239-246

[11] Jurič B. Matjaž, *The efficiency of distributed object models*, OOPSLA'99, ACM, New York, 1999

[12] Jurič B. Matjaž, Rozman Ivan, Heričko Marjan, Domajnko Tomaž, *CORBA, RMI and RMI-IIOP performance analysis and optimization*, World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, Vol. 8, Part 2, 2000, str. 582-587, vabljeno predavanje

[13] Object Management Group, The Common Object Request Broker: Architecture and Specification, Revision 2.4, 2000

[14] Soley R. M., Ph.D. (1995), Object Management Architecture Guide, OMG, John Wiley & Sons, Inc., Revision 3.0, Third Edition

[15] Vinoski Steve, *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*, IEEE Communications Magazine, Vol. 14, No. 2, February, 1997

◆

Dr. Matjaž B. Jurič je docent na Inštitutu za informatiko Fakultete za elektrotehniko, računalništvo in informatiko v Mariboru. Njegovo raziskovalno razvojno delo pokriva vse vidike objektne tehnologije s posebnim poudarkom na komponentnem razvoju programske opreme in distribuiranih objektnih sistemih. Sodeloval je pri razvoju RMI-IIOP, sestavnega dela Java 2 platforme. Med drugim je avtor devetih izvirnih znanstvenih člankov, samostojnega poglavja v knjigi, izdani pri založbi Cambridge University Press in recenzent revije Information and Software Technology Journal.

◆

Dr. Ivan Rozman je redni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Je vodja Inštituta za informatiko in dekan fakultete. Področja njegovega dela vključujejo programsko inženirstvo, objektno tehnologijo, vodenje projektov in kakovost v programskem inženirstvu. Je avtor več knjig in številnih znanstvenih ter strokovnih člankov.