UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

PRIMERJAVA TEMELJNIH PRISTOPOV PRI RAZVOJU INFORMACIJSKIH REŠITEV ELEKTRONSKEGA POSLOVANJA

Ljubljana, marec 2005 **DARKO GRADIŠNIK**

IZJAVA

Študent Darko Gradišnik izjavljam, da sem avtor tega magistrskega dela, ki sem ga napisal pod mentorstvom prof. dr. Mira Gradišarja in skladno s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne: 15. marec, 2005

Podpis: _____

KAZALO

1. UVOD

1.1. Uvod v magistrsko delo

Elektronsko poslovanje sega v začetek sedemdesetih let. V tem času so si lahko takšen način poslovanja privoščile samo velike korporacije, finančne ustanove in redka podjetja. Sledi obdobje izmenjave elektronskih podatkov EDI, tehnologija, ki omogoča elektronsko izmenjavo dokumentov, kar je pomenilo razširitev elektronskega poslovanja na veliko več gospodarskih subjektov. Prihod informacijske dobe spremljajo številni pojavi, ki so posledica svetovnih gospodarskih in finančnih tokov, koncentracij kapitala, spremenjenih političnih razmerij ter družbenih in institucionalnih sprememb v posameznih državah (Boar, 1997).

S komercializacijo interneta smo priča neverjetnemu razvoju elektronskega poslovanja, ki ima vpliv na poslovne modele, upravljanje oskrbnih verig in osnovna področja poslovanja. Pojavljajo se nove oblike poslovanja z namenom optimiziranja proizvodnje, prilagajanja kupcu in povečevanja konkurenčne prednosti. Vpliv interneta pa postaja posebno pomemben na področju elektronskega poslovanja. Pred desetimi ali petnajstimi leti nihče ni znal predvideti, da bo internet postal glavni distribucijski kanal za izdelke in storitve. Vse to je temeljito spremenilo ekonomijo, tržišče in podjetniško strukturo, izdelke in storitve; segment potrošnikov, vrednotenje in obnašanje kupcev. Toda vpliv interneta bo celo večji na socialnem in političnem področju, kot način kako bomo sprejemali svet in samega sebe (Drucker, 2002).

Podjetja, ki poslujejo na ustaljeni način so pred veliko preizkušnjo, ali bodo znala uspešno odgovoriti na te nove izzive. Potrebno je uporabiti možnosti, ki jih ponujajo sodobne informacijske tehnologije. Podjetja bodo morala ponovno opredeliti vizijo, cilje in strategije. Prilagajanje in nenehno izboljševanje poslovnih procesov velikokrat ne zadošča več. V določenih primerih bo potrebno v celoti prenoviti poslovne procese, kar pa lahko predstavlja velike stroške in rizike. Kaže se potreba po strateškem načrtovanju poslovne informatike in s tem nujnost spremembe načina vodenja, organizacije, socialnih in drugih sprememb. Turban (2003) pravi, da je pri uvajanju elektronskega poslovanja skoraj vedno potrebno izvesti Business Process Re-engineering (BPR).

Internet omogoča nekaj bistvenih prednosti pred ostalimi mediji. Zaradi razpoložljivosti, dostopnosti in interaktivnosti, je še posebno zanimiva informacijska arhitektura, ki temelji na spletnih tehnologijah. Ta tehnologija lahko zagotavlja univerzalen dostop do poslovnih informacij in aplikacij, ki jih potrebujejo kupci, dobavitelji, distributerji in ostali člani vrednostne verige. Učinkovito upravljanje oskrbnih verig je bilo pred desetletjem bolj ali manj teoretični proces, danes pa postaja eno najučinkovitejših konkurenčnih orodij. To ni zgolj tehnologija, temveč poslovna strategija, ki daje in odpira nove, učinkovitejše možnosti poslovanja (Kalakota, 2001).

Veliko podjetij se v ta namen odloča za portalske rešitve, ki morajo zadovoljiti pričakovanja po kakovostnih informacijah. Za uspešno elektronsko poslovanje in rabo interneta v sodobnem poslovnem svetu je zelo pomembno, da odgovorni za cilje in strategijo poznajo dovolj dobro tehnike in orodja, ki so pri tem na voljo. Pogosto se dogaja, da informatiki določajo cilje in strategije, ker se spoznajo na to tehnologijo, čeprav bi morali pri tem sodelovati le kot svetovalci. Upoštevati je potrebno tudi osnovna pravila in obnašanja ter zakonitosti, ki jih elektronski medij ponuja ali zahteva. Stewart (1997) verjame, da ekonomska moč velikih korporacij sloni na intelektualnih zmogljivostih in ne v osnovnih sredstvih.

Uporabniki spletnih strani so vse bolj zahtevni, želijo na enem mestu veliko kvalitetnih in svežih informacij. Rešitev teh zahtev je v uporabi spletnih storitev, ki omogočajo uporabo različnih virov informacij, programskih orodij, vse skupaj pa je lahko neodvisno od računalniške platforme.

Na tržišču vlada zelo močna konkurenca na področju razvojnih aplikacijskih orodij. Glavna igralca sta Microsoft z .NET in Sun Microsystems z J2EE tehnologijo. Vsak ima svojega paradnega konja in oba vlagata zelo veliko v marketinško promocijo svojih programskih orodij.

Cilj magistrske naloge je primerjava temeljnih pristopov pri razvoju informacijskih rešitev v elektronskem poslovanju. Raziskovanje temelji predvsem na proučevanju literature tujih in domačih avtorjev. Magistrsko delo sem razdelil na pet poglavij.

V začetnem delu sem opisal različne modele elektronskega poslovanja. Nakazujem prednosti in omejitve v elektronskem poslovanju. Opisujem zakonodajo na tem področju in potrebne ukrepe, ki jih morajo organizacije izvesti, če želijo uspešno vpeljati elektronsko poslovanje. V nadaljevanju sem opisal odprti internetni standard XML, ki omogoča strukturirano zapisovanje podatkov. V četrtem poglavju sem analiziral spletne storitve in jih prikazal na praktičnem primeru.

Jedro magistrske naloge predstavljata peto in šesto poglavje, kjer opisujem arhitekturo J2EE in .NET. V tem delu primerjam J2EE in .NET, poskušam določiti kriterije, ki bi jih bilo potrebno upoštevati pri izbiri določene tehnologije, ko se odločamo za elektronsko poslovanje.

2. ELEKTRONSKO POSLOVANJE

Živimo v digitalni dobi v kateri se način kako živimo in poslujemo drastično spreminja. Pritiski konkurence silijo podjetja v razmislek o inovativnih pristopih in uporabi sodobnih informacijskih tehnologij, ki podjetjem omogočajo primerjalno prednost na trgu. Elektronsko poslovanje ob dobrem poznavanju sodobnih tehnologij odpira nove poslovne horizonte, ki jih je nujno potrebno upoštevati pri snovanju poslovne strategije. Elektronsko poslovanje postaja nova paradigma, na podlagi katere bodo lahko podjetja izboljšala poslovanje. Zaradi intenzivne uporabe informacijskih tehnologij, bodo meje med prej neodvisnimi industrijami počasi izginjale, prišlo bo do konvergence med posameznimi sektorji.

Informacijska tehnologija postaja v globalnem trgu eden izmed najpomembnejših dejavnikov. Razvoj informacijske tehnologije in komunikacij še ne kaže znakov umirjanja (Gradišar 2003, str. 3). Ta tehnologija nima alternative, kar pomeni, da je trenutno edina smer in prihodnost, ki prinaša s seboj veliko prednosti in težav. Poganja jo nova ekonomija znanja, ki lahko zagotovi ogromen potencial, rast in nove zaposlitve. Nova ekonomija (digitalna ekonomija[1]) omogoča cenejši in lažji dostop do podatkov, kar ustvarja neslutene priložnosti za izdelavo novih proizvodov in storitev. Na ta način se ustvarja nova industrija, ki spreminja ostale in globoko vpliva na življenje državljanov.

Primerno razvita telekomunikacijska infrastruktura predstavlja osnovni pogoj za uspešno razvijanje informacijskih storitev. Zato je potrebno vlagati v razvoj telekomunikacij in poskrbeti za ustrezno deregulacijo telekomunikacij. Prehod v informacijsko družbo ne zahteva posebnega državnega usmerjanja. Dovolj je, da država poskrbi za osnovne pogoje delovanja.

Kljub napovedim, da bo ta tehnologija vplivala na moč vodilnih in vodstvenih delavcev, torej krepitev centralizacije, se je zgodilo ravno nasprotno. Danes smo priča decentralizaciji v organizacijah, ki jo povzroča decentralizacija informacijskih sistemov. Delavci na nižjih položajih imajo bistveno več informacij kot v preteklosti. To je posledica nizke cene in enostavnosti uporabe računalniške tehnologije. Podjetja imajo svoje oddelke povezane z računalniškimi mrežami, kar omogoča povezovanje posameznih informacijskih sistemov v povezan sistem in s tem usklajeno delovanje organizacije.

Z uporabo interneta se lahko podjetja povezujejo v preskrbovalne verige in s tem bistveno zmanjšajo stroške, istočasno se pa lahko hitreje in racionalneje odzivajo na spremembe v poslovnem svetu. Učinkovito obvladovanje oskrbnih verig s pomočjo novih tehnologij in njim

[1] Digitalna ekonomija (imenovana tudi internetna ekonomija, nova ekonomija ali spletna ekonomija) je ekonomija, osnovana na digitalnih tehnologijah, ki zajemajo digitalna komunikacijska omrežja (internet, intranet, ekstranet, itd.).

prilagojenim poslovnim procesom bo v naslednjih letih zelo pomembna naloga v podjetjih. Pogoj za uspeh so informacijsko povezani in pregledni poslovni procesi znotraj podjetja. To je pogoj, da se lahko podjetje poveže tudi z zunanjimi poslovnimi partnerji in z njimi načrtuje povezovanje poslovnih procesov. Po trditvah različnih avtorjev prihajamo v obdobje, ko se konkurenčnost ne meri več med posameznimi poslovnimi partnerji, ampak med oskrbnimi verigami podjetij.

2.1. Prenova poslovanja

Tradicionalni poslovni procesi so neprimerni za elektronsko poslovanje. Organizacije bodo morale v celoti prenoviti informacijsko tehnologijo in s tem povezane poslovne procese. Seveda to ne pomeni, da bo potrebno zavreči vse pretekle dobre rešitve. V določenih primerih se bodo organizacije raje odločile za postopno prenovo. Uvajanje elektronskega poslovanja v organizacijo lahko predstavlja tudi določeno stopnjo tveganja, kajti odločiti se je potrebno velikokrat tudi za nove in nepreizkušene poslovne modele. Prav zastareli poslovni procesi in odpor posameznih struktur v podjetju so lahko največja ovira za razvoj elektronskega poslovanja.

Čeprav se organizacije na splošno zavedajo pridobitev, ki jih lahko dosežejo s prenovo poslovnih procesov, so določene organizacije mnenja, da za to še ni pravi čas, ni posebne potrebe in da je izziv elektronskega poslovanja ter za to potrebne prenove poslovnih procesov še daleč. V določenih primerih so pomisleki upravičeni, še zlasti, če se organizacije otepajo z velikimi izgubami. Na žalost se tudi mnogo uspešnih podjetij ne odloči za zagon projekta prenove. Vzroki za to niso v bojazni pred neuspehom, kot bi se pričakovalo, temveč v tem, da ne čutijo potrebe ali pa je v podjetjih premalo odločnosti za tako korenite spremembe.

V prihodnosti bodo preživele le organizacije, ki bodo znale prilagoditi svojo proizvodnjo in storitve individualizirati in personalizirati. Individualizacija in personalizacija bosta temeljito spremenile način izvajanja celotnega procesa. Pri tem lahko odigra pomembno vlogo informacijska tehnologija, ki pa jo je potrebno uporabiti pravilno, kajti nepravilna uporaba lahko pripelje do delnih rešitev, ki v splošnem dajejo zelo slabe rezultate.

Prenova poslovanja zastopa strategijo korenite “prevetritve” obstoječih poslovnih pravil in predstavlja določeno stopnjo tveganja, ki je odvisna od naslednjih ključnih dejavnikov (Kovačič 2002, str. 24):

- motivacije - vodstvo organizacije mora zaupati in verjeti, da le celovita prenova poslovanja prinaša prednost pred konkurenco;
- vodenja projekta - odgovornost za vodenje in uspeh projekta mora prevzeti vodja, ki je član najožjega vodstva;
- zaupanja pri srednjem in vodilnem kadru;

- vizije – na novo opredeljeni strateški cilji morajo biti predstavljeni v obliki, ki je razumljiva in sprejemljiva za vse udeležence projekta;
- usmeritve - projektne aktivnosti in viri, ki so potrebni za izvedbo sprememb, morajo biti usmerjeni zlasti v spremembe k najpomembnejšim ciljem organizacije;
- opredelitve vlog in odgovornosti - vloge udeležencev projekta morajo biti opredeljene dosledno in podrobno pred izvedbo prenove poslovanja in po njej;
- merljivih rezultatov - rezultati prenove morajo biti konkretni, kot na primer na novo opredeljeni strateški cilji organizacije;
- tehnološke podpore - pri tem gre za uporabo metod in orodij, potrebnih za izvedbo prenove ter za izgradnjo informacijskega sistema, namenjenega informatizaciji prenovljenega poslovanja;
- strokovnega usmerjanja - svetovalno delo strokovnjakov s tega področja ne sme biti le nadzorno, temveč dejavno, pomeniti mora neposredno udeležbo na projektu;
- prevzemanja tveganja - vodstvo projekta se mora zavedati visoke stopnje tveganosti projekta in biti pripravljeno nase prevzeti tudi vse morebitne posledice.

Prenova poslovnih procesov

Prenova poslovnih procesov (BPR[2]) je v zadnjem času zelo aktualna tema. BPR predstavlja korenite spremembe v poslovanju. Z BPR želimo doseči pozitivne rezultate, kot so zniževanje stroškov, povečanje kakovosti izdelkov in storitev, skrajšanje dobavnih rokov in podobno. Poslovni proces opredeljujemo kot skupek povezanih izvajalskih in nadzornih postopkov, katerih posledica oziroma izid je načrtovan izdelek ali storitev (Kovačič 2002, str. 26). BPR je potrebno obravnavati v povezavi z vsemi drugimi dejavniki, ki sestavljajo socio-tehnični okvir organizacije.

2.2. Opredelitev elektronskega poslovanja

Pojem elektronsko poslovanje je širok izraz in različni avtorji ga različno pojmujejo. V slovenskem jeziku poznamo samo izraz – elektronsko poslovanje. V angleškem jeziku se uporabljata dva izraza: e-commerce in e-business. V zadnjem času se vse bolj uveljavlja izraz e-business, ki označuje elektronsko poslovanje v širšem smislu.

Po mnenju evropske unije je elektronsko poslovanje katerikoli oblika poslovanja, pri katerem stranke delujejo elektronsko, namesto da bi delovale fizično oziroma bi bile v neposrednem fizičnem stiku (Gradišar 2003, str. 21; povz. po Uporabna informatika 1997).

[2] Izraz "prenova poslovnih procesov" se je najprej pojavil na raziskovalnem področju v devetdesetih letih pod kratico BPR (Business Process Redesign) v raziskovalnem programu MIT (Massachusetts Institute of Technology).

Turban (Turban 2003) opredeljuje elektronsko poslovanje kot proces nabave, prodaje ali izmenjave, servisa in pridobivanja informacij prek računalniških mrež. Večina elektronskega poslovanja se izvaja prek interneta

Elektronsko poslovanje ni nič novega. Veliko podjetij v avtomobilskem in prodajnem sektorju je uporabljalo elektronsko poslovanje v obliki elektronske izmenjave podatkov EDI (Electronic Data Interchange) že pred dobrimi dvajsetimi leti. Informacijska tehnologija je sčasoma postala nosilni dejavnik poslovanja. Elektronsko poslovanje ne predstavlja samo prodajanje ali nakupovanje v elektronski obliki temveč tudi način izboljšanja produktivnosti, doseganje novih strank in izmenjavo znanja. Velika podjetja so razvila svoje strateške informacijske sisteme za podporo elektronskemu poslovanju.

2.3. Poslovni modeli

Poslovni model je metoda, ki opisuje, kako naj podjetje posluje, da bo proizvajalo prihodek, ki ga potrebuje za svoj obstoj. Poslovni modeli so lahko enostavni ali pa tudi zelo kompleksni. Primer kompleksnega modela je na primer internetna televizijska postaja, ki zastonj oddaja svoj program. Njeno preživetje je odvisno od kompleksnega modela, ki združuje mnogo različnih aktivnosti, kot so oglaševanje, oblikovanje vsebine, itd.

Obstaja mnogo poslovnih modelov. Timmers (Turban 2003, str.14; povzeto po Timmers 1999) opredeljuje sedemnajst poslovnih modelov, od tega so najpogostejši naslednji:

- Imenuj-svojo-ceno (Name-your-price): Model, ki so ga oblikovali pri Priceline.com, dopušča kupcu, da sam določi ceno, ki jo je pripravljen plačati za nek izdelek ali storitev.
- Dinamično posredništvo (Dynamic brokering): V digitalni dobi lahko stranke specificirajo zahteve katerim mora ustrezati izdelek ali storitev, ki jih zanima. Te zahteve se posredujejo prek interneta ponudnikom zahtev, z avtomatičnim vabilom za sprejem ponudbe.
- Obratne dražbe (Reverse auctions): Veliki kupci (zasebni ali državni) uporabljajo za svoje nakupe sistem ponudb. Ponudbe lahko sedaj kupec opravlja kar preko interneta, pri čemer prihrani na času in denarju. Več vladnih ustanov po svetu se je že odločilo za sistem elektronskih ponudb kot edinega načina za odločanje o nakupih. Obratne dražbe bistveno zmanjšujejo administrativne stroške.
- Pridruženo trženje (Affiliate marketing) je poslovna ureditev, v kateri tržni partnerji na svojo spletno stran postavijo oglasno pasico (banner) neke družbe, kot je na primer Amazon.com. Vedno, ko stranka klikne na pasico in se s tem premakne na spletno stran oglaševalca, oglaševalec plača provizijo podjetju - gostitelju.
- Skupinski nakupi (Group purchasing): Cena na enoto je običajno manjša, kadar kupujemo več enot hkrati. Popusti so namreč običajni pri količinskih nakupih. Z uporabo koncepta skupinskih nakupov, lahko tudi manjša podjetja ali celo posamezniki, dobijo popust. Elektronsko poslovanje omogoča s konceptom elektronske agregacije skupinske nakupe.

Neodvisno podjetje oziroma neodvisna stranka zbere več manjših naročil, jih poveže v večje naročilo in se z dobaviteljem pogaja za čim boljše pogoje plačila.

- E-tržnice in borze (E-marketplaces and exchanges): E-tržnice so kot izolirane aplikacije obstajale že desetletja. Nekatere borze vrednostnih papirjev so popolnoma računalniško podprte že od 1980. Dandanes e-tržnica uvaja operativno učinkovitost v trženje in če je dobro organizirana in vodena, lahko predstavlja prednost tako za kupca kot tudi za prodajalca. Posebno zanimanje vlada za navpične tržnice (vertical marketplaces), ki se osredotočajo na eno panogo.

Poslovni model je mogoče celo patentirati. Kot vsa ostala tržišča ima tudi internet več tržnih sektorjev, vendar je internet edinstven po tem, da so vsa ta tržišča virtualna. Zaradi tega se poslovni modeli v fizičnem in virtualnem prostoru razlikujejo. Načrtovanje poslovnega modela predstavlja strateško orožje in tržno razlikovanje.

2.4. Zakonodaja na področju elektronskega poslovanja

Priča smo velikemu številu virusov in vdorov v računalniške sisteme. Zato je varnost oziroma zaščita podatkov ključnega pomena pri prehodu na elektronsko poslovanje. Eden izmed varnostnih elementov so tudi standardi in zakonski predpisi, ki urejajo elektronsko poslovanje z ustrezno pravno podlago.

Slovenija je leta 2000 sprejela zakon o elektronskem poslovanju in elektronskem podpisu (ZEPEP[3]-uradni list RS št. 57/00), ki predstavlja nadgradnjo obstoječi zakonodaji in osnovo za e-poslovanje. S tem so odpravljene ovire, ki so jih postavljale pravne norme elektronskemu poslovanju, zasnovane in sprejete v času papirnatega poslovanja.

Zakonodaja mora biti jasna, predvidljiva, podpirati mora konkurenčnost ter jasno mora določati ravnotežje med svobodo izražanja, varovanjem osebnih in javnih interesov s posebnim poudarkom na varstvu potrošnika. Vzpostaviti se mora takšen pravni okvir, ki mu bo potrošnik zaupal, poslovni svet pa bo spodbujal k vlaganju in razvoju (Jerman Blažič 2001, str. 162).

Prvi zakon o elektronskem podpisu je bil sprejet že leta 1995 v ZDA. Prvi evropski zakon o elektronskih podpisih pa je sprejela Nemčija leta 1997. V letu 2001 je UNCITRAL[4] sprejel nov modelni zakon o elektronskih podpisih, ki definira uporabo elektronskih podpisov za potrebe komercialnih aktivnosti in podobno kot direktiva evropske unije se osredotoča predvsem na sproščanje ovir pri elektronskem poslovanju v pogojih, ko se zahteva podpis na podatkih, ki jih stranke pošiljajo na elektronski način (Jerman Blažič 2001, str.114).

[3] ZEPEP - zakon o elektronskem poslovanju in elektronskem podpisovanju.

[4] UNCITRAL Model Law on Electronic Signatures, United nations Commission on Internet trade Law.

Podpis oblikovan na elektronski način ima dokazno vrednost. To pomeni, da se takšni obliki podpisa v postopkih ne sme odreči veljavnost. Elektronski podpis lahko oblikujemo z uporabo različnih elektronskih pripomočkov. Zakon ne določa, da mora biti podpis oblikovan z uporabo sredstva za varno podpisovanje, ki ga opredeljuje drugi člen ZEPEP. Zakonodajno urejanje elektronskega poslovanju posega tudi v druge zakone in podzakonske predpise, kot so zaščita zasebnosti in osebnih podatkov, intelektualne lastnine in varstva potrošnika.

2.5. Prihodnost elektronskega poslovanja

Nekateri strokovnjaki napovedujejo, da bo elektronsko poslovanje v nekaj letih doseglo takšne meje, da več ne bomo govorili o elektronskem poslovanju, ker bo vso poslovanje potekalo elektronsko. Z uveljavljanjem interneta in vse cenejšim dostopom do elektronskih storitev, lahko pričakujemo vedno večje prihodke z elektronskim poslovanje. Organizacije bodo v bodoče morale prilagoditi svoje informacijske sisteme. Spremembe v informacijskih sistemih se bodo odražale tudi v spremembi organiziranosti, kar bo vplivalo tudi na strukturo organizacije.

Pričakujemo lahko, da se bodo tradicionalne oblike poslovanja ohranile le v redkih primerih, zelo specifičnih tržnih segmentih ali protekcionističnih okoljih. Kupec bo vedno bolj obveščen in zato tudi zahteven. Izdelek ali storitev bo želel imeti v zelo kratkem času, obravnavati pa ga bo potrebno kot posameznika in ne kot del množice. Ponudba podjetij bo morala biti zelo prilagodljiva, s katero se bo podjetje približalo kupčevim potrebam. Uspela bodo podjetja, ki bodo znala uporabiti sodobno tehnologijo in s tem ponuditi edinstvene izdelke in storitve po sprejemljivi ceni.

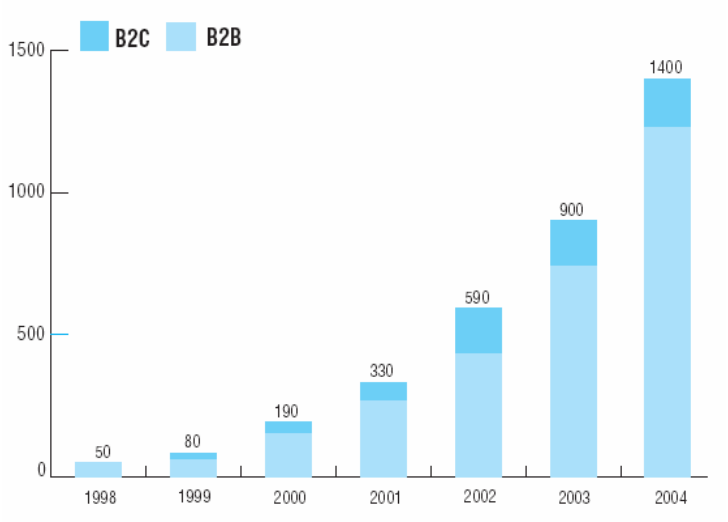
Inovativnost bo postala odločilnega pomena. Podjetja bodo iskala nove poslovne modele in prilagajale svoje informacijske sisteme za povečanje primerjalne prednosti. Vse to bo zahtevalo veliko prilagodljivost poslovnih sistemov. Ocenjuje se, da bodo večja podjetja manj prilagodljiva zaradi svoje velikosti. Mala podjetja se soočajo s pomanjkanjem finančnih virov. Elektronsko poslovanje bo lahko predstavljalo rešitev tudi za manjša podjetja, ki bodo lahko z dobro načrtovano informacijsko tehnologijo konkurirala tudi večjim podjetjem. Poslovni modeli bodo postajali vse bolj zapleteni, z veliko stopnjo integracije poslovnih procesov in tesnem poslovnem sodelovanju med člani vrednostne verige, zato bo te modele vedno težje posnemati. Pričakujemo lahko, da bo v prihodnosti poslovni model postal strateško orožje za doseganje konkurenčne prednosti.

Kljub določenim popravkom napovedi obsega elektronskega poslovanja v preteklosti, se je elektronsko poslovanje ponovno konsolidiralo. Po zadnjih ocenah znaša promet približno 1400 milijard dolarjev v letu 2004 (Berryman in Hech, 2003). Pri ocenah, koliko je doseženega prometa z elektronskim poslovanjem, je potrebno upoštevati tudi nevidni del, na katerega ima

elektronsko poslovanje posredni vpliv. Glavni del elektronskega poslovanja poteka med podjetji B2B, približno sedemdeset odstotni.

Po ocenah Gartner Group, je elektronsko poslovanje med podjetji prešlo v peto generacijo (Turban 2003, str. 217). Ta generacija vključuje sodelovanje med dobavitelji in kupci, interno in eksterno vrednostno verigo.

Slika 1: Obseg elektronskega poslovanja v milijardah dolarjev



Vir: Keenan Vision

2.6. Omejitve v elektronskem poslovanju

Netehnološke omejitve

Uvedba elektronskega poslovanja v organizaciji zahteva spremembe v organizacijski strukturi in v procesih, ki so ključni. Zahteva uvedbo strategije nove ekonomije, vse to pa zahteva spremembo mišljenja tako vodstva organizacije kot vseh ostalih zaposlenih. Kot osnovno omejitev uvedbe elektronskega poslovanja pa lahko razumemo nezaupanje vodstva v partnerja s katerim naj bi poslovali na elektronski način. Partnerja s katerim poslujemo ne vidimo, ne poznamo in z njim po navadi ne pridemo v stik drugače kot prek žice. Veliko pomislekov v organizacija velja tudi elektronskemu plačevanju in njegovi varnosti, zaradi možnih vdorov in zlorab.

2.7. Izmenjava elektronskih podatkov (RIP)

Internet omogoča podjetjem neposredno in transparentno komunikacijo. Svetovni splet sestavljajo programska oprema, nabor protokolov in dogovorjenih pravil. Internet in na internetu temelječe tehnologije zagotavljajo naslednje:

- univerzalne in hitre povezave (komunikacijski kanal),
- poceni in brezpapirno komuniciranje,

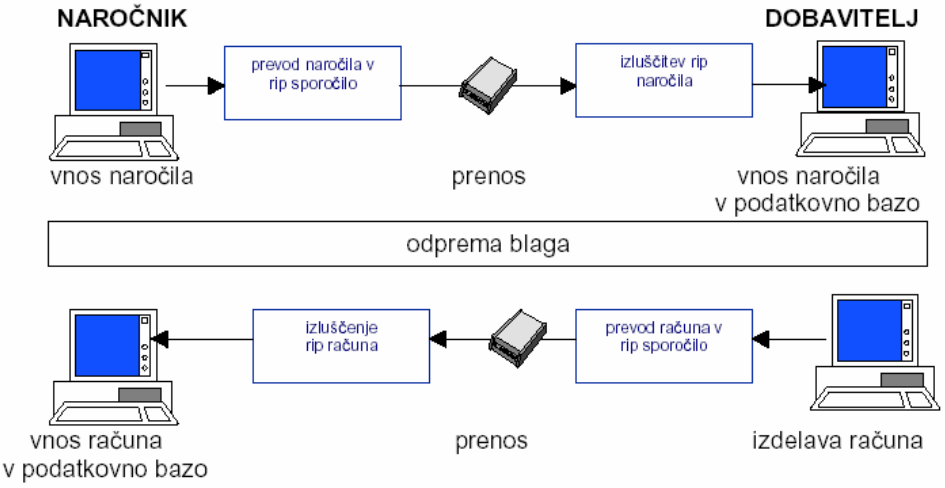
- poceni in brezpapirno izvajanje poslovnih transakcij,
- od računalniške strojne in sistemske opreme neodvisno programsko opremo ter storitve.

Mnoga podjetja so hitro spoznala prednosti, ki jih nudi internet. Z njim lahko bistveno izboljšajo tudi nabavni proces v podjetju. Podjetja so se zaradi tega povezala v organizacije, ki sestavljajo konzorcij. Rezultat takih povezovanj so določeni odprti standardi za elektronsko poslovanje na internetu, ki jih bom opisal v nadaljevanju.

2.7.1. Računalniška izmenjava podatkov

RIP[5] omogoča izmenjavanje podatkov v standardizirani obliki in tako omogoča avtomatizacijo postopkov njihove obdelave oziroma komunikacijo računalniških rešitev brez posredovanja človeka. RIP tehnologija uporablja natančno določeno obliko sporočila neodvisno od računalnikov, ki so vpleteni v izmenjavo informacij. Informacije lahko izmenjavamo med različnimi programskimi platformami. V preteklosti je bila prevladujoča oblika elektronskega povezovanja po omrežjih VAN[6].

Slika 2: Računalniška izmenjava podatkov



Vir: Šafarič 2000, EAN Slovenija

Računalniška izmenjava podatkov poteka na izvajalnem nivoju, ki oblikuje bazo podatkov organizacije. Računalniško izmenjavanje dokumentov je možno le, če so izpolnjeni naslednji pogoji:

- sodelujoče organizacije se morajo dogovoriti za obliko in vsebino elektronskih sporočil (določiti standard),

[5] RIP – Računalniška izmenjava podatkov po mednarodnem standardu. V angleščini se uporablja kratica EDI – Electronic Data Interchange.

[6] VAN – Value Added Networks (omrežja z dodano vrednostjo)

- zagotoviti programsko opremo, ki omogoča prevajanje sporočil iz standardizirane oblike v obliko, ki je primerna za obdelavo podatkov z računalniškim sistemom posamezne organizacije,
- zagotoviti ustrezen komunikacijski kanal.

Organizacije se morajo med seboj dogovoriti za enotne šifrante materialov, izdelkov in storitev. Če organizacija posluje z več partnerji, bo morala tak dogovor skleniti z vsako organizacijo posebej. Vsaka organizacija bo tako imela lahko različne oblike dokumentov, ki pa so vsebinsko enotni. V realnosti je tako dogovarjanje lahko zamudno, drago in vir mnogih napak. Zato so se izoblikovali posebni standardi posamezno za vsako panogo v gospodarstvu[7]. V tem primeru se pred pošiljanjem dokumenta (Slika 2) s posebnim prevajalnikom najprej pretvori v standardno obliko. Na strani prejemnika pa se s pomočjo podobnega prevajalnika dokument avtomatično prevede v obliko, ki jo informacijski sistem prepozna in podatke vpiše v bazo. Omenjeni način prenosa podatkov lahko na primer uporabimo za samodejno naročanje blaga. Vpletenost človeka v proces naročanja in pošiljanja blaga se zmanjša na minimum, kar pomeni manj napak in manjše stroške. Na trgu je mnogo programskih rešitev in paketov, ki omogočajo vedno lažje in cenejšo povezovanje z RIP-om in tako postajo dostopno tudi manjšim podjetjem.

Poleg komunikacijskega medija so za RIP nujno potrebni tudi standardi za računalniško izmenjavanje podatkov. Hkrati z uporabo interneta za poslovanje med organizacijami se hitro razvijajo tudi nove tehnologije. Te omogočajo nove in še tesnejše poslovne povezave, tako pa se vedno bolj povečuje tudi potreba po standardih, ki bi omogočili preprost in učinkovit način RIP-a prek interneta (Global Commerce Internet Protocol 2001). V ta namen so se oblikovale različne pobude, skupine in združenja. Trenutno sta najbolj dejavni delovni skupini Global Commerce Initiative (GCI) in ebXML.

2.7.2. Stroški in koristi RIP-a

Računalniška izmenjava podatkov prinaša podjetjem številne prednosti, njena uporaba pa prinaša tudi določene stroške. RIP podjetjem omogoča, da zmanjšajo administrativne stroške, odpravo nekaterih nepotrebnih aktivnosti oziroma večkratni vnos podatkov, ter s tem tudi zmanjša možnost napak pri vnosu. Administrativni stroški se zmanjšajo tudi zaradi elektronskega prenosa podatkov med partnerji. RIP omogoča tudi povečanje učinkovitosti ter racionalizacijo poslovanja, saj se tokovi med poslovnimi partnerji avtomatizirajo, kar podjetjem omogoča obvladovanje večjih količin podatkov, brez povečanja števila zaposlenih oziroma celo z njihovim zmanjšanjem. Med posredne učinke RIP-a lahko štejemo še skrajšanje časa od naročila do realizacije, skrajšanje dostavnih časov, zmanjšanje zalog in prihranek

[7] Vlagajo se veliki napori v oblikovanje svetovnih standardov na posameznih področjih gospodarstva. Na primer avtomobilska industrija.

denarja (Bračko, 1997, str. 23). Mnogokrat pa je vzrok za uvedbo RIP-a predvsem nuja, da lahko podjetje sploh ostane konkurenčno. Stroški RIP-a so (Bračko, 1997):

- stroški analize obstoječega poslovanja,
- stroški izobraževanja in usposabljanja,
- stroški svetovanja,
- stroški zaposlenih,
- stroški programske in strojne opreme,
- telekomunikacijski stroški,
- stroški vzdrževanja sistema.

Pri ocenjevanju koristi in stroškov RIP-a se je treba zavedati, da se koristi pojavijo na dolgi rok. Zato je analizo koristi in stroškov nujno izvajati na dolgi rok, saj so skoraj vsi stroški v zvezi z uvedbo RIP-a v podjetju enkratni, tako da jih je potrebno razdeliti na daljše obdobje.

2.7.3. Standard UN/EDIFACT[8]

Leta 1986 se je v okviru Organizacije združenih narodov pričel razvoj mednarodno priznanega standarda računalniške izmenjave podatkov, imenovanega UN/EDIFACT. Standard podaja sintaktična pravila za pripravo sporočil, ki se izmenjujejo med partnerji po elektronski poti. Pomembna lastnost UN/EDIFACT standarda je v tem, da je oblikovan tako, da je v celoti neodvisen od strojne in programske opreme, sporočilo oziroma dokumenti pa so v celoti šifrirani. Slovnico tvorijo podatkovni elementi, ki so zapisani v standardu ISO 9735, besedni zaklad pa tvorijo podatkovni elementi, ki so predstavljeni v standardu ISO 7372.

Prizadevanja potekajo v smeri, da bi vsaka posamezna država jasno opredelila vse potrebne papirne in elektronske postopke ter dokumente, objavila kriterije elektronskega poslovanja v državi ter oblikovala nacionalni standard elektronskih sporočil, ki bi bil v okviru celotnega standarda (nekakšen podstandard UN/EDIFACT).

Ministrstvo za znanost in tehnologijo je leta 1991 ustanovilo tudi Urad za UN/EDIFACT, ki naj bi zagotavljal povezavo s standardom in omogočil vpliv na njegovo oblikovanje ter prilagajanje nacionalnim potrebam. Urad za UN/EDIFACT je potreben, saj je slovensko gospodarstvo še prešibko, da bi lahko posamezna podjetja sodelovala s svojimi strokovnjaki v delovnih skupinah pri Organizaciji združenih narodov.

[8] UN/EDIFACT je standard, ki vsebuje skupek svetovno sprejetih standardov in navodil za računalniško izmenjavo podatkov, še posebej v panogah povezanih s trgovino med neodvisnimi z informacijsko tehnologijo podprtimi podjetji.

2.7.4. Pomanjkljivosti RIP-a

RIP se je najprej uveljavil v zaprtih omrežjih, pozneje pa je bil prilagojen tudi za uporabo na internetu. Prevladujoča standarda na tem področju sta ANSI X12 in EDIFACT. Razvoj RIP-a omejujejo naslednji dejavniki (Roman 1999, str. 615):

- Najete vode ponujajo komercialni ponudniki, cena teh vodov je pa ponavadi zelo visoka. RIP si lahko privoščijo samo velika podjetja. Zaradi tega ponavadi ni mogoče povezati vse udeležence v oskrbni verigi.
- Izvajanje poslovnih transakcij preko VAN omrežij je zamudno in neučinkovito. To je zaradi tega, ker so VAN omrežja prilagojena za občasni podatkovni prenos in so neprimerna za procesiranje podatkov.
- Zaradi pomanjkanja odprtih standardov, se morajo udeleženci prilagoditi strukturi podatkov, ki jo predpišejo velika podjetja, kar omejuje možnosti za izmenjavo elektronskih podatkov z drugimi podjetji.
- RIP je standard, ki je neprilagodljiv, neroden in nerazširljiv. Kljub temu število uporabnikov RIP-a narašča. Razlog je v tem, da gre za preizkušeno tehnologijo in večina podjetij ne želi tvegati z novejšo tehnologijo.

3. Označevalni jeziki: SGML, HTML in XML

3.1. SGML

SGML (Standard Generalized Markup Language) je meta-jezik, ki služi za opis (pretežno) besedilnih dokumentov. Kot standard je bil sprejet že leta 1986, in ima bogato zgodovino uporabe. Konec šestdesetih let je Stanley Rice, oblikovalec knjig iz New Yorka, predlagal nabor značk za opis zgradbe spisov. Ta nabor je bil ob podpori združenja »Graphic Communications Association« razvit v prvi opisni nabor »GenCode«. Na teh osnovah so leta 1969 Charles Goldfarb, Edvard Mosher in Raymond Lorie za podjetje IBM ustvarili GML (Generalized Markup Language). Vanj so vpeljali tudi pojem zvrsti spisa in gnezdenja sestavin. GML je postal osnova IBM-ove programske podpore založništva. Goldfarb je nadaljeval raziskave o zgradbi spisov in razvil vrsto dodatnih sestavin.

Leta 1978 je bila pri ANSI (American National Standard Institute) ustanovljena skupina za pripravo standarda jezika za opis besedil, ki bi izhajal iz jezika GML. Prvi osnutek standarda je bil predstavljen leta 1980; šesta različica pa je že dobila vlogo industrijskega standarda (GCA 101–1983). Leta 1984 sta pričela s sodelovanjem ANSI in ISO (International Standards Organization) na pripravi mednarodnega standarda. Osnutek je bil objavljen leta 1985, naslednje leto pa tudi sam standard SGML (ISO 8879:1986 Information processing – Text and office systems – Standard Generalized Markup Language).

SGML zagotavlja naslednje:

- ločitev opisa zgradbe od opisa oblike prikaza – znakovna podatkovna baza;
- uporabo gradiv v različne namene, vključevanje dodatnih informacij;
- prenosljivost: neodvisnost od vrste računalnika (znakovni opis v izbrani kodi – praviloma koda ASCII);
- povečanje učinkovitosti priprave, uporabe, vzdrževanja in razpečevanja informacij;
- trpežnost in trdoživost gradiv: nevezanost na posameznega proizvajalca programske opreme;
- odprtost: pestrost orodij različnih proizvajalcev;
- nedvoumnost, zmanjševanje števila napak v podatkih;
- izboljšan nadzor: povezave med posameznimi sestavinami, omejen dostop do posameznih podatkov.

3.2. HTML

HTML (HyperText Markup Language) se uporablja na WWW (World-Wide Web) od leta 1990 dalje. To je preprost jezik za označevanje, ki temelji na SGML in omogoča pripravo sestavkov (besedilo, slika, zvok, video, itd.) prenosljivih med različnimi vrstami računalnikov. HTML je v trajnem razvoju, za katerega skrbi IETF (Internet Engineering Task Force). V današnji rabi so naslednje različice: HTML 1.0, HTML 2.0, HTML 3.2 in HTML 4.0. Prvo različico je izumil Tim Bearnes Lee. Pri svojem delu je veliko uporabljal internet za raziskovanje in komuniciranje. Ogrodje HTML 1.0 jezika je prevzel iz SGML in s tem poenostavil jezik, saj je HTML jezik primeren za širšo uporabo. Jezik HTML 1.0 je imel samo pet elementov: <TITLE>, <P>, ločitev glave - <HEAD> od telesa <BODY>, ter uporabo GIF datotek.

HTML 2.0 je bil razvit leta 1995 pod iniciativo IETF (RFC1866). Poleg že naštetih elementov HTML 1.0 omogoča še <INPUT>, ki služi za dodajanje gumbov, gesel in teksta, <SELECT>, <TEXTARE>, ki omogoča pisanje besedila v več vrsticah
 in <P>.

Leta 1995 je bila narejena HTML 3.0 verzija. Podpirala je GIF datoteke, če brskalnik ni podpiral slik se je namesto slike pojavil tekst (<ALT>). Poleg tega je vseboval še oznake <IMG>, <HR>, dodajanje, brisanje, ter podčrtavanje besedila ter možnost določanja ozadja (<BgColor>).

HTML 3.0 različica se ni obnesla in to kar je omogočila, je vsebovala različica HTML 3.2. Ta je bila oblikovana kot enoten HTML standard, leta1997 pod okriljem W3C (World Wide Web konzorcij), ki je poskrbel tudi za njegov nadaljnji razvoj in zamenjala je različico HTML 2.0. Dodali so ji možnost izbire raznih barv, animacij in dodajanje zvoka, ki zelo poživijo videz strani. Poleg tega pa vsebuje še oznake <SCRIPT> in <STYLE>.

Zadnja različica je HTML 4.0, ki je bila standardizirana leta 1999. Omogoča okvirje (<FRAMES>), uporabo CSS (Cascading Style Sheets), ki definira kako naj brskalnik prikaže HTML elemente, <OBJECT> zamenja <IMG>, itd. Različica tega standarda je zadnja, nadalje so v veljavi standardi XHTML, DHTML, XML, ki prevzemajo vlogo HTML.

3.3. Primerjava SGML, XML in HTML

XML je skrajšana verzija SGML standarda, ki omogoča procesiranje na tisoče različnih tipov dokumentov. SGML je zelo dober označevalni jezik, vendar preveč kompleksni za uporabo v svetovnem spletu. Sama specifikacija vsebuje več kot petsto strani. XML delovna skupina je zmanjšala obseg specifikacij na samo šestindvajset strani, pri tem pa je ohranila petindevetdeset odstotkov funkcionalnosti. Razvijalci jezika XML so izločiti iz XML vse ukaze, ki niso potrebni za delo s svetovnim spletom. Specifikacija XML vsebuje tudi dodatne funkcije, kot na primer XSL[9] in XSLT[10].

XML v primerjavi s HTML nima v naprej definirane množice oznak, ampak omogoča dodajanje lastnih oznak, ki pa morajo biti v dokumentu urejene glede na ustrezna pravila. Razširljivost predstavlja veliko prednost, saj nam omogoča definiranje novih slovarjev glede na potrebe konkretne aplikacije, gospodarske panoge ali načina komunikacije med poslovnimi partnerji. V XML so podatki ločeni od svoje predstavitve, to pomeni, da ta jezik služi le za definicijo vsebine, kako pa bo grafično prikazan, pa je odvisno od vmesnika, ki ga uporabljamo. Iste podatke je mogoče prikazati na več načinov. Način predstavitve lahko spreminjamo neodvisno od sistema, ki ga uporabljamo. Podatke v XML lahko uporabi katerikoli programska rešitev, ki podpira jezik XML. Dokument XML je neobčutljiv na nadgradnjo programske in strojne opreme.

XML je zasnovan na 16-bitni kodi Unicode sistema, HTML in SGML pa temeljita na ASCII standardu (American Standard Code for Information Interchange). Zatorej lahko z XML uporabljamo tudi tuje slovnice, ne pa samo angleško, ki je osnova pri ASCII standardu.

HTML je sicer odličen za predstavitev določene vsebine na svetovnem spletu, vendar ne olajša integracije informacij na spletu z obstoječimi poslovnimi aplikacijami. V prihodnosti HTML standard verjetno ne bo izpodrinjen. Bo pa deloval v sodelovanju z ostalimi standardi (XML, PHP...). Še vedno je zelo praktičen in enostaven za oblikovanje preprostih spletnih elementov. V uporabi bo, glede na zahtevnost spletnih vsebin.

---

[9] XSL (angl. Extensible Style Language) predstavlja jezik za oblikovno predstavitev XML dokumentov.

[10] XSLT (angl. Extensible Stylesheet Language) predstavlja jezik za transformacijo XML dokumenta, ki omogoča preoblikovanje, dodajanje in brisanje oznak in lastnosti.

3.4. XML

XML (Extensible Markup Language) je programski jezik, ki se uporablja za opisovanje strukture in pomena podatkov. Osnovni standard oziroma priporočilo W3C je »XML 1.0 Recommendation«, ki je bilo izdano leta 1998. Po začetni počasni osvojitvi znotraj internetne skupnosti, je XML postal »de facto« jezik za komunikacijo med internetnimi brskalniki, računalniki, strežniki in aplikacijami.

XML je nastal leta 1995, kot predlog konzorcija World Wide Web (W3C). Cilj je bil razviti enostaven jezik, podoben enostavnosti jezika HTML, ki je prav zaradi enostavnosti toliko razširjen. Pri tem so bili zelo uspešni. Oblikovalcu dokumenta omogoča, da dokument oblikuje po svoji zamisli. Leta 1998 je izšla verzija XML 1.0, ki so ga podprla vsa pomembna podjetja, kot so: SAP, IBM, Oracle, Microsoft, itd. Na temelju osnovne specifikacije XML jezika, se je razvila množica tehnologij in specifikacij, ki širijo in izboljšujejo uporabnost XML-a na številnih področjih uporabe.

XML je torej označevalen jezik, kar pomeni da besedilu iz dokumenta doda določen podatek, s čimer označi to besedilo. Besedilo se nato na zaslonu ali tiskalniku prikaže skladno z oznakami. Dokument tako obsega vsebino in označevalce, ki določajo, kakšno vlogo ima ta vsebina (npr. vsebina v naslovu dokumenta ima drugačen pomen kot vsebina v opombah). Vnaprej določenega nabora označevalcev ni, zato tudi ne more biti nobene vnaprej določene vsebine. Vsa vsebina dokumenta XML je določena s programsko rešitvijo, ki dokument obdeluje, ali z vnaprej pripravljeno predlogo. Z izrazom dokument pa ne mislimo le na običajna besedila v dokumentih, temveč tudi veliko drugih strukturiranih podatkov, kot so vektorske grafične slike, poslovna sporočila, matematične enačbe, objekti, programski vmesniki itd. Obstajajo označevalni jeziki za različna področja, kot so: Mathematical Markup Language, CML (Chemical Markup Language) in WML (Wireless Markup Language).

3.5. XML sintaksa

Dokument mora biti napisan po točno določenih pravilih. XML razlikuje med malimi in velikimi črkami. Naslednja stvar, ki jo je potrebno upoštevati, je prisotnost korenskega elementa, znotraj katerega morajo biti vsi ostali. Še ena od lastnosti XML-ja je striktnost pri upoštevanju znakov – iz tega izhaja, da se ohranijo vsi presledki v tekstu, atributne vrednosti pa morajo biti obvezno v narekovajih, ki so lahko enojni ali dvojni, razen kadar imamo znotraj atributne vrednosti že dvojne narekovaje. Takrat je potrebno uporabiti enojne. Pri snovanju oznak je potrebno upoštevati nekaj pravil (W3Schools, 2003):

- imena zaznamkov lahko vsebujejo črke, številke in ostale znake;
- imena zaznamkov se ne smejo začeti s številko ali ločilom;
- prav tako se ne smejo začeti s kombinacijo znakov XML;
- niti ni dovoljeno vanje vstavljati presledkov.

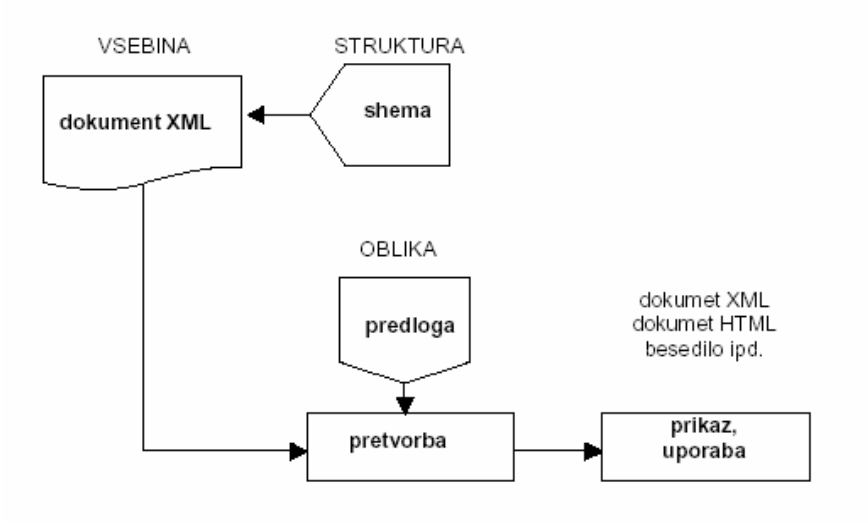
Slika 3: Primer računa v XML obliki

```
<?Xml version=”1.0”?>
<račun>
<glava>
<številka>123/2002</številka>
<prejemnik>Telekom Slovenije</prejemnik>
<naslov>Cigaletova</naslov>
</glava>
<postavke>
<postavka>
<naziv>osebni računalnik</naziv>
<znesek valuta=”SIT”>2.000.000</znesek>
</postavka>
<postavka>
<naziv>tiskalnik</naziv>
<znesek valuta=”SIT”>900.000</znesek>
</postavka>
</postavke>
</račun>
```

3.6. Shema XML

Dokument XML vsebuje poleg podatkov tudi njihovo strukturo, ki pa ni natančno določena (npr. niso navedeni tipi podatkov, ponavljanje). Za elektronsko poslovanje je pomembna uporaba skupnega jezika, zato je W3C razvil priporočila za shemo XML (XML Schema). Shema XML je nekakšen slovar, ki natančno opisuje podatke v dokumentih XML. Uporablja slovnico XML, podpira različne podatkovne tipe in omogoča izgradnjo novih podatkovnih tipov. Je nadgradljiva in dopušča znotraj sheme naslavljanje tudi drugih shem. S shemo XML se preverjajo vsebina in zaporedje podatkov ter pravilnost in veljavnost dokumenta XML. Dokument, ki ustreza glavnim pravilom priporočila XML, je pravilen. Dokument, ki je skladen z ustrezno shemo XML, pa je veljaven. Izraz shema je prevzet iz relacijskih baz podatkov in na področju XML pomeni model oziroma vzorec za veliko dokumentov (npr. račun, dobavnico). Shemo XML lahko razumemo tudi kot dogovorjen skupen slovar, ki določa ureditev dokumentov in ki ga pri izmenjavanju dokumentov uporabljajo računalniški programi. Iz izvorne oblike lahko dokument XML za predstavitev ali uporabo v drugi organizaciji preslikamo v primerno obliko s pomočjo jezika XSL (Extensible Stylesheet Language).

Slika 4: Pretvorba dokumenta v XML



Vir: Harold,2001

Z jezikom XSL izdelamo predloge, s katerimi določimo preslikave in s tem oblikujemo dokument XML. Dokument XML lahko pretvorimo v dokument HTML, tekstovno datoteko, drug dokument XML in podobno ali v obliko, ki jo lahko bere človek ali računalniški program. Pravilnost vsebine dokumenta in veljavnost dokumenta zagotovimo s shemo XML.

3.7. XML in varnost

Zelo pomembni vidik elektronskega poslovanja je varnost. Varnostne storitve v elektronskem poslovanju obsegajo (Jerman Blažič; 2001):

- overjanje identitete;
- zagotavljanje zaupnosti in neokrnjenosti informacij;
- nadzor dostopa;
- preprečevanje zanikanja sporočil in aktivnosti;
- omogočanje stalne razpoložljivosti.

Podatke lahko varujemo na različne načine, odvisno od zahtevane stopnje zaščite in strukture sistema. V elektronskem poslovanju se najpogosteje uporablja kriptografija in iz nje izpeljane oblike zaščite (npr. elektronski podpis). Pri kriptografiji gre za šifriranje sporočil. Šifrirni algoritmi preoblikujejo podatke dokumenta v tajne, neberljive. Pri obratnem procesu pa tajne podatke preoblikujejo v berljive (Gradišar, 2003). Tudi pri uporabi tehnologije XML se lahko uporabljajo standardne metode zaščite, poleg tega pa obstajata pri konzorciju za svetovni splet dva standarda: XML (XML Signature) in šifriranje XML (XML Encryption). V tem kontekstu naj omenim še t.i. kanonično obliko dokumenta XML (Canonical XML), ki med transakcijami zagotavlja logično istovetnost dokumentov.

3.7.1. podpis XML

Definira sintakso in pravila za kreiranje in predstavitev elektronskih podpisov v XML. Lahko je v obliki ovojnice na podatkih znotraj istega dokumenta XML, lahko pa je ločen od vsebine, ki ji pripada. Elektronsko podpisani del dokumenta je znotraj zaznamka tipa <SignedInfo>, ki vključuje tudi podatke o vrsti uporabljenega algoritma, medtem ko podatke o ključu, ki je potreben za validacijo dokumenta najdemo znotraj zaznamka <KeyInfo> (ta element ni obvezen) kot kaže Slika 5. Informacije, ki so elektronsko podpisane se nahajajo med vrsticami s02 in s12:

Slika 5: Primer elektronsko podpisanega XML dokumenta

```
<Signature Id="MyFirstSignature"
      xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/
         REC-xml-c14n-20010315"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/
         xmldsig#dsa-sha1"/>
<Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
<Transforms>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
           20010315"/>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/
           xmldsig#sha1"/>
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
```

Vir: IBM, Enabling XML Security.

3.7.2. šifriranje XML

Namen šifriranja XML dokumenta ni vedno v tem, da prikrijemo vse podatke v dokumentu. Smiselno je omogočiti šifriranje samo tistih delov dokumentov za katere je to potrebno. Na primer imamo šifrant strank, ki vsebuje imena, naslove in številke kreditnih kartic. Zaposleni ima pravico vpogleda do vseh podatkov z izjemo številk kreditnih kartic. Aplikacija tako temu zaposlenemu dešifrira le imena in naslove, medtem ko mu številk kreditnih kartic ne dešifrira in ne prikaže. Podatki, ki so znotraj šifriranega zaznamka, se »ovijejo« z novim zaznamkom tipa <EncryptedData...>, ki ga dopolnjujeta še <CipherData> in <CipherValue>. Slika 6 prikazuje primer šifriranja XML dokumenta, kjer je šifriran podatek o številki kreditne kartice.

Slika 6: Primer šifriranja XML dokumenta

```
<?xml version='1.0'?>
    <PaymentInfo xmlns='http://example.org/paymentv2'>
      <Name>John Smith<Name/>
      <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
       xmlns='http://www.w3.org/2001/04/xmlenc#'>
         <CipherData><CipherValue>A23B45C56</CipherValue></CipherData>
      </EncryptedData>
    </PaymentInfo>
```

Vir: IBM, Enabling XML Security.

4. SPLETNE STORITVE (WEB SERVICES)

Spletne storitve predstavljajo zadnji dosežek v razvoju aplikacij. Privlačne so ra razvijalce na vseh platformah. Osnovni koncept spletnih storitev je preprost – omogočajo nam oddaljeno klicanje procedur (RPC[11]) prek omrežja. Ne gre za prvo tehnologijo, ki bi nam to omogočala, vendar se razlikuje od predhodnih tehnologij po tem, da uporablja platformsko neodvisne standarde. Uporabnik spletne storitve mora poznati URL storitve, podatkovne tipe, ki se uporabljajo za klicanje procedur, ni pa potrebno poznati ali je spletna storitev zgrajena v Javi in teče na Linux operacijskem sistemu, ali je zgrajena na osnovi ASP.NET, ki teče na Windows operacijskem sistemu.

Spletno storitev lahko definiramo kot del poslovne logike, ki se nahaja nekje na omrežju, dosegljiv s protokolom HTTP ali SMTP. Glede na to definicijo, bi lahko v skupino spletnih storitev klasificirali že določene obstoječe tehnologije, vendar jih ne. Te tehnologije vključujejo win32, J2EE, CORBA in CGI. Glavna razlika med temi tehnologijami in tehnologijo spletnih storitev, je v njihovi standardizaciji.

Spletna storitev je podobna storitvi v resničnem, fizičnem svetu. Programske rešitve so po definiciji storitve: ponujajo odgovor na težave, avtomatizirajo opravila, olajšajo vsakodnevne naloge itd. Pojem dostavljive programske opreme je lahko nastal šele z rojstvom spleta. Zaradi široke sprejemljivosti in dostopnosti je idealna infrastruktura za dostavo programskih rešitev.

[11] RPC (Remote Procedure Call) – predstavlja klicanje oddaljenih procedur. Pomeni zmožnost pognati podprogram, ki je realiziran na oddaljenem računalniku. Razvijalec do oddaljenih podprogramov praviloma dostopa prek aplikacijskega vmesnika API.

Osnovni tehnologiji interneta sta HTTP (Hyper Text Transfer Protocol) in HTML (Hyper Text Markup Language). Vodilna podjetja, ki razvijajo programske rešitve, so določila potrebne tehnologije za spletne storitve (konzorcij za svetovni splet – W3C[12]):

- XML (eXtensible Markup Language),
- WSDL (Web Services Description Language),
- SOAP (Simple Object Access Protocol),
- UDDI (Universal Description, Discovery, and Integration).

Poslovni pomen spletnih storitev

Spletne storitve za poslovne namene lahko bistveno zmanjšajo stroške poslovanja organizacij. Odpirajo nove možnosti tudi za manjša podjetja. Uporaba spletnih storitev v organizaciji zahteva prilagoditev in preoblikovanje poslovnih procesov. Spletne storitve omogočajo tudi integracijo obstoječih rešitev. Tudi starejše aplikacije lahko predstavimo kot spletne storitve. Ustvarimo lahko npr. vmesni sistem, integriran s sistemom v ozadju, ki prikazuje svoje funkcije kot eno ali več storitev prek spleta. Storitev je lahko brezplačna, namenjena javnosti za plačilo ali pa je preprosto mišljena kot storitev za partnerska podjetja. Ideja, ponuditi rešitev kot storitev, ki jo je mogoče dostaviti kadar koli, kamor koli in v katero koli napravo, je privlačna. Ne samo, da lahko spletne storitve poenostavijo prenos podatkov znotraj organizacije, ampak lahko z njimi tudi zapolnimo luknjo v prihodkih od propada številnih podjetij .com. Storitev, ki prej ni dobro delovala prek interneta zaradi tega, ker je bilo vedno potrebno uporabnikovo posredovanje, sedaj nenadoma deluje brez tega. Poleg tega lahko integriramo spletne storitve v tradicionalne aplikacije. Iz svojih spletnih komponent lahko izdelamo rešitve ali pa popestrimo različice v svoji liniji izdelkov tako, da nekatere funkcije spremenimo v storitve za plačilo.

4.1. Komponentne arhitekture

Novejše programske aplikacije praviloma uporabljajo specializirane programske dodatke, ki jih imenujemo komponente. S komponentami lahko popestrimo aplikacijo, saj omogočajo razvijalcu programske opreme razvoj uporabniško prijaznih aplikacij. Komponente so postale osnovni gradnik sodobnih aplikacij. Neločljivo so povezane z večslojnimi arhitekturami, objektnim razvojem, pa tudi spletnimi storitvami, XML in celo integracijo aplikacij. Komponente najdemo na vseh slojih, vendar danes poseben pomen pripisujemo poslovnim in spletnim komponentam. Ponavadi vključujemo v aplikacijo komponente različnih proizvajalcev. Problem pa nastane, ko želimo komponente uporabiti v omrežju. Rešitev tega problema predstavlja prav spletna storitev.

[12] Če želite izvedeti več o konzorciju World Wide Web, obiščite njihovo stran na naslovu www.w3c.org.

Osnovna prednost komponentnih arhitektur izhaja iz enkapsulacijske[13] tehnike, ki jo uporabljajo aplikacije. Vsebina komponente je navzven popolnoma zakrita s programskim vmesnikom. Komunikacija med komponento in njenim okoljem poteka, podobno kot pri objektni tehnologiji, le preko določenih vmesnikov oziroma metod. Zaradi ločenosti programskega vmesnika od notranje programske strukture so komponente izjemno prilagodljive in enostavne za vzdrževanje.

Komponente nikoli ne nastopajo samostojno, temveč so vključene v aplikacije. Arhitekture s komponentami zagotavljajo robustnost, združljivost in zanesljivost poslovnih aplikacij. V primerjavi z ostalimi arhitekturami potrebujejo manj sistemskih in omrežnih virov. Poslovne aplikacije so zaradi visoke učinkovitosti komponent hitrejše in stabilnejše pri velikem številu odjemalcev.

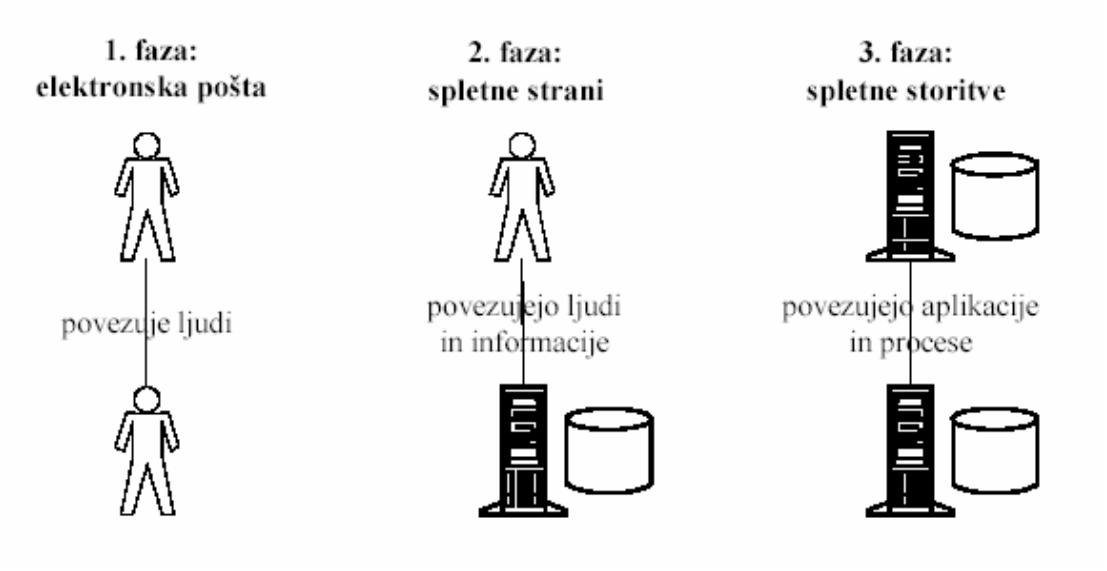
4.2. Poslovno povezovanje z elektronskim poslovanjem

Zaradi velika števila različnih povezav med podjetji, ne more nobeno podjetje vsiljevati svoje tehnologije drugim. Če imamo enostavni tip povezovanja (eden-z-enim), cena eksponentno narašča s številom povezav. Zato je primerna rešitev, ki omogoča ceneno, dinamično in standardizirano poslovno povezovanje. Internet predstavlja decentralizirano globalno povezovanje, zato je primeren tudi za poslovno povezovanje, ker izpolnjuje naslednje zahteve (Durchslag 2001, str. 8):

- povezovanje heterogenih sistemov, ki so med seboj enakovredni;
- prožne dinamične povezave s številnimi poslovnimi partnerji;
- preproste programske komponente;
- visoka zanesljivost rešitve, ki upošteva padajoče inkrementalne stroške ob povečanju števila povezav;
- robustno povezljivost, ki temelji na izmenjavah sporočil;
- razločno delitev funkcionalnosti namesto tesne vertikalne povezanosti enega samega proizvajalca;
- možnost najema funkcionalnosti, ne da bi bilo zato potrebno postaviti lastno infrastrukturo in razviti lastne aplikacije;
- neomejenost povezovanja zaradi odvisnosti od odprtih standardov.

[13] Enkapsulacija (angl. Encapsulation) ovijanje, zastiranje. Princip pri razvoju računalniških aplikacij, kjer izpostavljeni programski vmesnik zakrije notranjost komponente ali objekta.

Slika 7: Razvojne stopnje elektronskega povezovanja



Vir: Wong, 2001

Spletne storitve spadajo v tretjo fazo elektronskega povezovanja. Predstavljajo tehnološko osnovo za vrsto modularnih, dinamično sestavljenih poslovnih aplikacij, ki temeljijo na odprtih standardih interneta. Spletne storitve se bodo hitro uveljavile v poslovnih aplikacijah, saj temeljijo na obstoječi tehnologiji in ne zahtevajo večjih dodatnih vlaganj. Dostopne so preko uveljavljenih internetnih protokolov v obliki programskih komponent, ki nudijo povezavo do obstoječih poslovnih aplikacij in prenašajo njihovo funkcionalnost na internet. Te komponente lahko dinamično sestavljamo glede na zahtevano funkcionalnost. Spletne storitve bodo s časoma, zaradi cenene možnosti povezovanja, zabrisale tradicionalne meje med podjetji.

4.3. Primeri uporabe spletnih storitev

Spletne storitve predstavljajo novost v razvoju aplikacij. Pritegnile so pozornost in interes razvijalcev programske opreme ne glede na katerih platformah razvijajo programske rešitve. Ne gre za prvo tehnologijo, ki bi omogočala RPC, vendar je to prva tehnologija, ki uporablja platformo nevtralnih standardov kot sta HTTP in XML. To nam omogoča, da so podrobnosti implementacije popolnoma nevidni uporabniku. Vse kar mora uporabnik vedeti je naslov spletne storitve in podatkovne tipe, ki se uporabljajo za klicanje metod. Ni pa potrebno vedeti ali je spletna storitev zgrajena z Javo in teče na operacijskem sistemu Linux, ali je zgrajena na osnovi ASP.NET tehnologije in teče na operacijske sistemu Windows. V prihodnosti lahko pričakujemo zelo veliko razširjenost spletnih storitev. Da je to trditev lažje razumeti si poglejmo uporabnost spletnih storitev na naslednjih primerih.

Predstavljajmo si, da smo založnik knjig. Dobili smo zahtevo od naše e-prodajalne po informacijah o določeni knjigi. Vzdrževalec spletne strani potrebuje podatke o vsebini knjige, sliko ovitka in ostale informacije, ki služijo boljši predstavitvi knjige. Kreiramo lahko novi

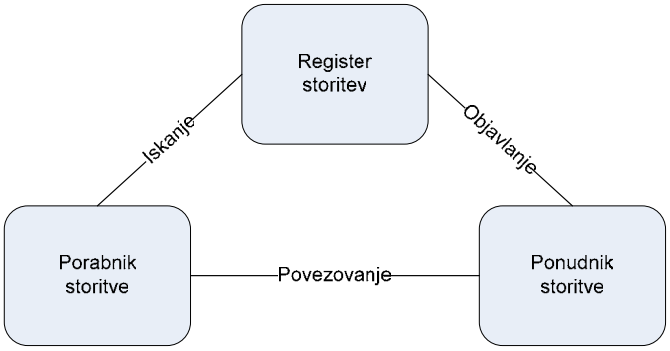
dokument v HTML obliki z vsemi temi podatki in pošljemo te podatke. Ta način pa ima eno pomanjkljivost. Potrebujemo namreč centralizirani mehanizem za distribucijo istih informacij na več naslovov. Prodajalec lahko zahteva sliko knjige z enostavno poizvedbo, ki bi lahko izgledala tako »pošlji sliko knjige ISBN=xxx«. To poizvedbo bi lahko implementirali s funkcijo fnOvitekISBN(številkaISBN). Ta funkcija bi vrnila sliko, ki bi jo lahko v e-prodajalni direktno vključili v svojo spletno stran. Programer internetne strani lahko vključi ime funkcije na mesto, kjer mora biti slika knjige. Na ta način imajo vsi e-prodajalci vedno sveže podatke od založnika knjig.

Naslednji primer bi lahko bila funkcija, ki vrne ceno artikla v realnem času. To lahko predstavlja za prodajalce izjemno konkurenčno prednost, kajti kupec je tako seznanjen s trenutnimi cenami in dodatnimi ugodnostmi. Zanimiv primer spletne storitve predstavljajo integrirani finančni servisi. Visoko varna internetna stran lahko zbira finančne podatke o bankah, borzah, finančnih novicah, sedanjo vrednost lastnega finančnega stanja itd. V bližnji prihodnosti lahko pričakujemo zelo veliko tovrstnih storitev.

4.4. Model spletne storitve

Običajno spletno storitev sestavljajo naslednje entitete: ponudnik storitev, porabnik storitev in register storitev.

Slika 8: Model spletne storitve



Vir: Basha, Jeelani S., 2002

Ponudnik storitev predstavlja entiteto, ki ponuja spletno storitev. Kot primer takšne storitve si lahko zamislimo veletrgovca, ki želi prodajalnam knjig omogočiti naročanje knjig preko interneta. Ponudnik te spletne storitve mora narediti dve stvari, če želi izkoristiti spletno storitev v polni meri. Najprej mora opisati spletno storitev v standardnem formatu, ki ga razumejo vsi uporabniki. Nato mora poskrbeti, da bo njegova spletna storitev poznana vsem potencialnim porabnikom teh storitev. Porabnik storitev pozna funkcionalnosti spletne storitve glede na opis, ki ga je napravil ponudnik storitev. Podrobnosti spletne storitve poišče porabnik storitve v registru. Bolj pomembno je to, da lahko porabnik storitve na tem mestu najde vse

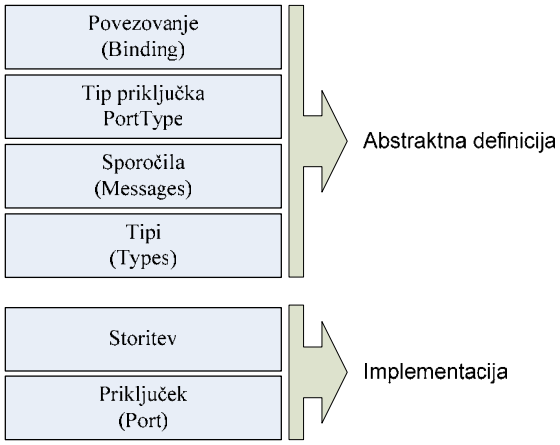
pomembne podatke, kot so: povezava podatkov za izmenjavo, uporaba metod in klicanje raznih procedur.

Register servisov predstavlja centralno lokacijo na kateri lahko ponudnik storitev objavi in porabnik storitev najde ustrezno spletno storitev. Tipično se na tem mestu objavijo informacije o podjetju, tehnični opis in detajli kako uporabiti spletno storitev. Pri običajni spletni storitvi potekajo naslednje tri operacije: iskanje, povezovanje in objava podatkov. Za izvajanje teh operacij je potrebna komunikacija med aplikacijami. Za vsako posamezno operacijo potrebujemo standard, s katerim lahko opišemo spletno storitev neodvisno od izbrane platforme in programskega jezika.

4.4.1. WSDL (Web Service Description Language)

WSDL jezik predstavlja mehanizem s katerim lahko ponudniki spletne storitve opišejo funkcionalnosti svoje storitve. Ta označevalni jezik služi kot storitvena pogodba (contract of the service). S posebno slovnico XML opiše strukturo spletne storitve in določi njeno »vstopno točko« - to je URL, kjer lahko drugi najdejo našo storitev. Naslednja slika prikazuje konceptualni pogled spletne storitve.

Slika 9: Konceptualni opis spletne storitve



Vir: Basha, Jeelani S., 2002

Definicija spletne storitve je v osnovi razdeljena na dva neodvisna dela: abstraktni opis in specifikacijo implementacije. Abstraktni del vsebuje naslednje elemente:

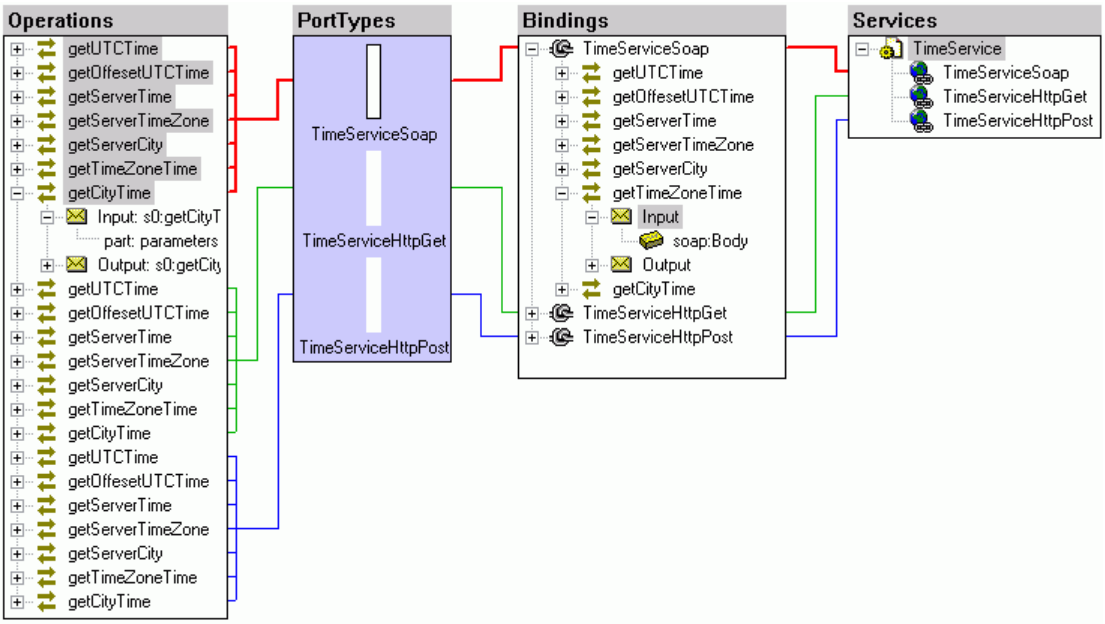
- **Port Type** - vsebuje nabor operacij, predstavljene kot operacijski elementi , ki so prisotni v spletni storitvi. Operacija lahko vsebuje vhodni ali izhodni podatek.
- **Messages** - predstavlja definicijo sporočil, ki se pošiljajo. Podobno kot parameter v klicanju metode.
- **Types** - vsebuje podatkovne tipe, ki so prisotni v sporočilu.

- **Bindings** - poveže elemente »Port Type« z določenim protokolom. Če želimo, da se določena operacija izvrši prek interneta, je potrebno uporabiti specifični transportni protokol s katerim pošljemo podatke.

WSDL urejevalnik

WSDL urejevalnik omogoča urejanje, vizualizacijo in validacijo WSDL datoteke. WSDL je zelo primeren kot vmesnik za definicijski jezik (IDL), ki se porablja za arhitekturo spletnih storitev. Naslednja slika prikazuje primer urejevalnika za WSDL.

Slika 10: WSDL urejevalnik



Vir: http://www.altova.com/features_wsdl.html

4.4.2. SOAP (Simple Object Access Protocol)

SOAP, ki je bil predstavljen leta 1998, je rezultat skupnega dela več podjetij (Microsoft, Developmentor, IBM, UserLand in mnogo drugih). SOAP 1.1 specifikacija je prva verzija, ki je bila predana W3C. Protokol je osnovan na metajeziku XML, uporablja pa se za podajanje informacij v eno smer. Informacije, ki se prenašajo v sporočilu SOAP, so lahko v obliki dokumenta ali klica za sprožitev oddaljenega postopka RPC. Eden izmed ključnih razlogov za današnjo veliko razširjenost tega protokola je v minimalističnem pristopu, ki so ga avtorji uporabili za njegov opis. Predvideli so le ogrodje in tako omogočili prilagajanje uporabe SOAP za specifične namene.

V porazdeljenem izvajanju računalniških opravil je eden od glavnih izzivov oddaljeno klicanje metod oz. storitev na oddaljenem računalniku. Na voljo je sicer nekaj rešitev, npr. DCOM (Distributed COM), ki pa so avtorsko zaščitene in zahtevajo veliko pasovno širino. SOAP pa reši te težave na odprt, preprosti način, zato je logična rešitev za uporabo v spletnih storitvah. Ker je SOAP izrazito enostranski protokol za sporočanje, se sporočila razlikujejo glede na to, ali so zahteve oziroma odgovori nanje. Poleg informativnih sporočil so lahko sporočila tudi napake pri sporočanju. Napaka je edina oznaka znotraj vsebine sporočila, ki jo protokol natančno določa.

Glavne prednosti SOAP-a so (Scott Short, 2002):

- Ni tesne povezave s programskim jezikom. Razvijalci programske opreme lahko uporabijo najnovejše programske jezike. Še posebno je dobrodošla neodvisnost programskega jezika pri obstoječih aplikacijah.
- Ni tesne povezave z določenim transportnim protokolom. SOAP specifikacija določa kako se mora SOAP povezati s HTTP ali SMTP protokolom. Toda SOAP sporočilo ni prav nič drugega kot XML dokument, ki ga lahko prenašamo z vsakim protokolom, ki podpira prenos znakov.
- Ni tesne povezave z nobeno objektno infrastrukturo. Večino distribuiranih objektnih sistemov lahko razširimo (nekateri so že) za podporo SOAP-a.

Osnovni namen razvoja SOAP specifikacije je v povezovanju že obstoječih standardov. Na primer, SOAP definira podatkovni tip znotraj XML sheme in ne določa s katerim protokolom bomo prenesli informacijo. SOAP lahko povežemo z obstoječim protokolom kot sta HTTP ali SMTP.

SOAP omogoča interoperatibilnost med različnimi okolji. Zgrajen je na obstoječih industrijskih standardih. Zato aplikacije, ki tečejo na platformah, ki podpirajo te standarde, lahko komunicirajo s SOAP sporočili, z aplikacijami, ki tečejo na drugih platformah. Na primer, namizna aplikacija, ki se uporablja na osebnem računalniku, lahko komunicira z zaledno aplikacijo, ki teče na glavnem računalniku (angl. mainframe), če le zmore sprejemati in pošiljati XML prek HTTP-ja.

4.4.3. UDDI (Universal Description, Discovery and Integration)

Ustvarjanje spletne storitve, ki jo je mogoče priklicati na daljavo prek omrežja, je le en del zgodbe. Najtežji del pride, ko skušajo razvijalci odkriti, katere spletne storitve so na voljo. Za ta namen uporabimo UDDI, ki predstavlja imenik spletnih storitev. Podjetjem je potrebno omogočiti oglaševanje in uporabnikom iskanje spletnih storitev. Obstajata dve vrsti mehanizma: UDDI in DISCO. UDDI predstavlja centralni hierarhično razporejeni imenik spletni storitev; DISCO pa je bolj svobodni način odkrivanja spletnih storitev.

Infrastruktura UDDI je sestavljena iz registra in registratorja. Register vsebuje vse podatke UDDI imenika, registrator pa opravlja UDDI registracijo v korist uporabnikov. Registrator je lahko ponudnik internetne storitve (ISP – internet Service provider), ponudnik B2B, individualno podjetje itd. V tem trenutku gostijo register Microsoft, IBM, SAP in Hewlett-Packard. Sad njihovega dela je UDDI, ki je platformno neodvisno in odprto ogrodje za opisovanje storitev, odkrivanje podjetij in integracijo poslovni storitev z uporabo interneta, pa tudi delujoči register, ki je na voljo že danes[14]. Na UDDI spletni strani (www.uddi.microsoft.com) lahko vnesemo ime svojega podjetja, stike po oddelkih, informacije o stikih, kratek opis svojega podjetja in seveda informacije o povezavah in spletnih storitvah, ki jih ponujamo. To so aktivne rumene strani spletnih storitev. Ker UDDI omogoča, da podjetja sama na splošno dostopen in standardiziran način opišejo svoje storitve, se lahko hitreje poiščejo in izvajajo elektronske transakcije med svojimi sistemi. Celotno spletno mesto Microsoft UDDI deluje z uporabo ASP.NET.

4.5. Primer spletne storitve

V tem delu bom na enostavnem primeru prikazal praktični primer spletne storitve. Izračunal bom obrok posojila glede na znesek kredita, obresti in način odplačevanja obrokov (na začetku ali koncu meseca). Za načrtovanje sem uporabil programsko orodje Visual Studio .NET 2002, programiranje pa sem izvedel s programskim jezikom Visual Basic .NET.

S programskega vidika gledano, predstavlja spletna storitev razred (Class), ki se nahaja na spletnem strežniku. Samo delovanje je podobno klasični ASP aplikaciji. Razlika je le v tem, da spletni strežnik v odgovoru na zahtevo ne kreira HTML strani, temveč se obnaša podobno kot funkcija: uporabnik pokliče funkcijo po imenu, vstavi potrebne parametre in kot odgovor dobi rezultat te funkcije. Rezultat je lahko število, tekst ali objekt.

Za programiranje spletne storitve ne potrebujemo posebnega dodatnega programskega znanja; če znamo napisati funkcijo v Visual Basicu, jo lahko napišemo tudi v spletni storitvi. Seveda je potrebno upoštevati, da v primeru, da se mora VB funkcija izvajati na ISS (Internet Information Services), mora biti rezultat funkcije zakodiran v primerno obliko za prenos prek HTTP protokola. Teoretično lahko napišemo takšno funkcijo celo v ASP-ju, vendar bi bilo to opravilo precej zahtevno. Veliko programerjev in managerjev (še zlasti pri Microsoftu) meni, da bodo spletne storitve povzročile revolucijo na področju spletnega programiranja. To je povsem mogoče. Gre namreč za dejstvo, da spletne storitve še zdaleč niso tako okorne, kot predhodni spletni razredi, ki so se lahko uporabljali pri programiranju ASP. Te razrede zaradi okornosti veliko programerjev sploh ni uporabljalo. Zato so ASP aplikacije raje razvijali s skriptnim jezikom VBScript, čeprav z veliko omejitvami.

[14] Več informacij o registru, lahko najdete na naslovu www.uddi.org/about.html. Prikaz ogrodja UDDI pa na naslovu www.uddi.microsoft.com.

Izgradnja spletne storitve

Čeprav lahko napišemo programsko kodo za spletno storitev v enostavnem urejevalniku teksta, nam lahko sodobno programsko orodje, kot je na primer Visual Studio, prihrani veliko časa.

Slika 11: Funkcija za izračun posojila

```
Imports System.Web.Services
<WebService(Namespace := "http://tempuri.org/")> _

Public Class Service1
   Inherits System.Web.Services.WebService

Public Function Posojilo(ByVal Znesek As Decimal, ByVal stMesecev As _ Integer,
ByVal Obresti As Decimal, ByVal PaciloZacMeseca As _ Boolean) As Decimal
      If Obresti < 0 Then
         Throw New Exception("Obresti ne smejo biti negativne!")
      End If
      If Obresti < 1 Then
         Obresti = Obresti / 12
      Else
         Obresti = Obresti / (12 * 100)
      End If
      Return Pmt(Obresti, stMesecev, Znesek, 0, PaciloZacMeseca)
End Function
End Class
```

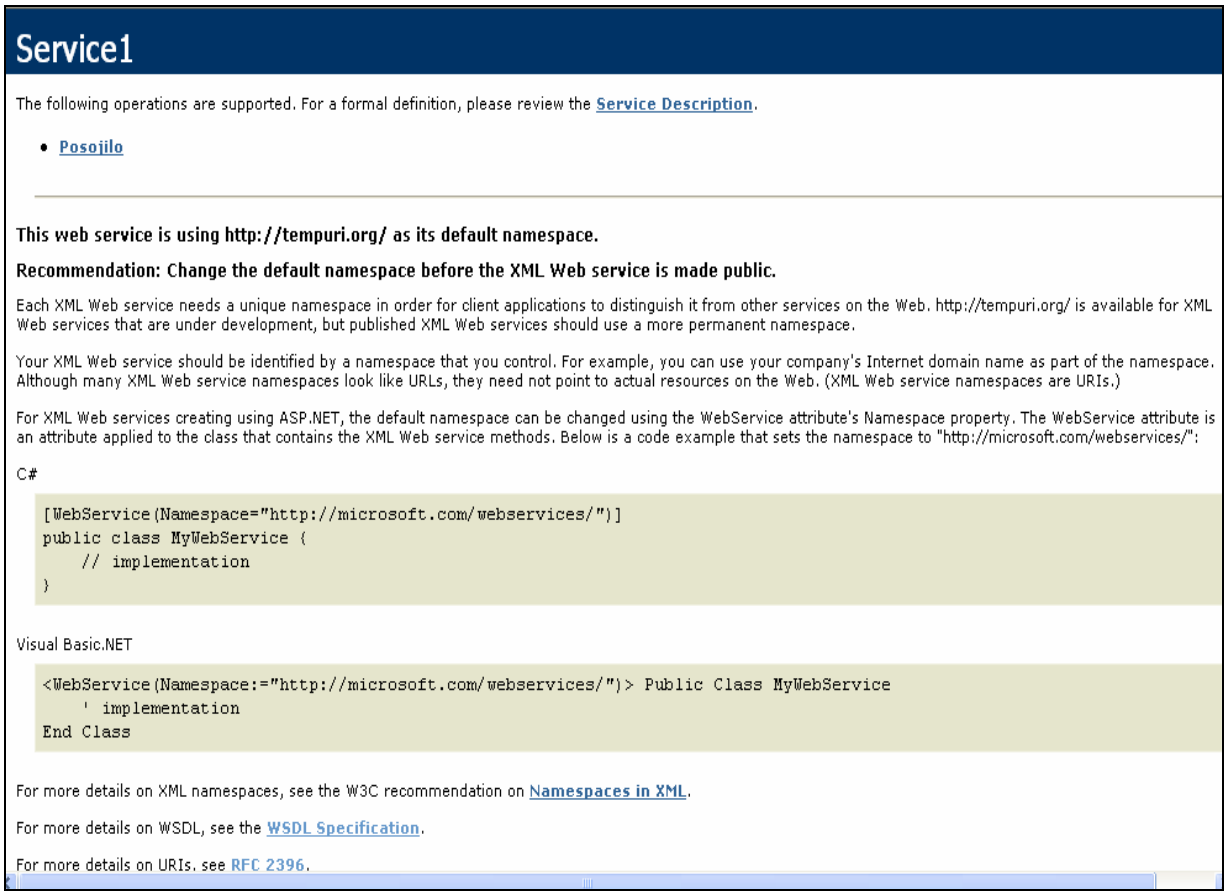
Če bi pisali v urejevalniku besedila, bi morali na začetku dodati naslednjo vrstico (Visual studio doda že sam):
<%@ webservice class=”Service1” language=”vb” %>

S tem stavkom določimo prevajalniku, da predstavlja preostanek kode spletno storitev, ki je napisana v programskem jeziku Visual Basic. Ime razreda je v zgornjem primeru »Service1«. V realnem primeru se ime določi na podlagi imena podjetja in namena servisa. Datoteko je potrebno shraniti s končnico ».asmx«.

Za testiranje spletne storitve je dovolj, da vpišemo v internetni raziskovalec URL http://localhost/Service1.asmx. Prikaže se nam internetna stran, ki je nismo kreirali sami (Slika 12). Ta stran se fizično ne nahaja na našem strežniku, temveč nam jo prevajalnik kreira

programsko, ko prvič prevedemo programsko kodo. Pod naslovom spletne storitve imamo seznam metod.

Slika 12: Testiranje spletne storitve

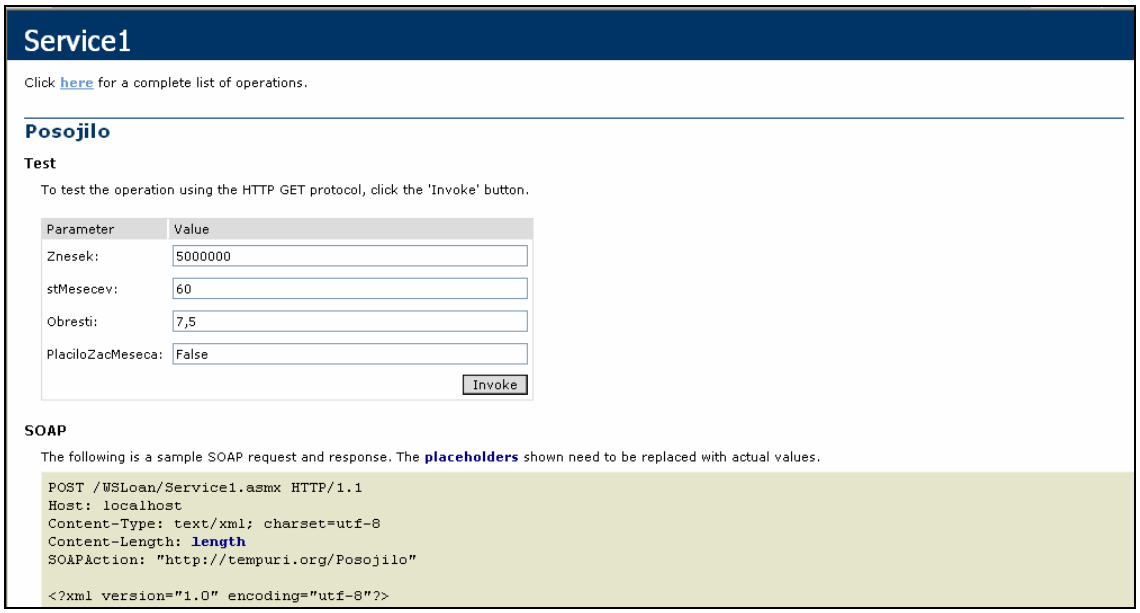


Na seznamu je samo ena metoda »Posojilo«. Če kliknemo na to metodo, bomo videli argumente na novi strani, kot prikazuje Slika 13. Na tej strani lahko preizkusimo spletno storitev. Vnesemo podatke v vsa štiri polja in kliknemo na gumb »Invoke«. Rezultat funkcije se nam prikaže na novi strani v obliki XML zapisa:

```
<?xml version="1.0" encoding="utf-8" ?>
<decimal xmlns="http://tempuri.org/">-100189.742978118</decimal>
```

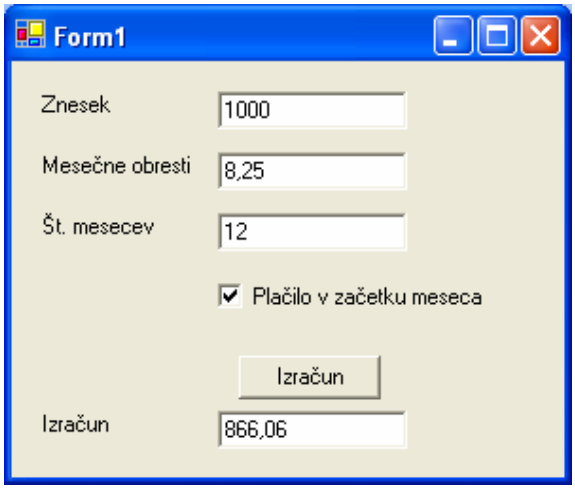
Rezultat je na prvi pogled nekoliko nenavaden, vendar je to edini način, da prenesemo informacijo od strežnika do odjemalca naloženo na HTTP protokol. Najlepše pri vsem tem je to, da ni potrebno instalirati nobenih dodatnih komponent na strežniku in tudi ne pri odjemalcu.

Slika 13: Testiranje funkcije



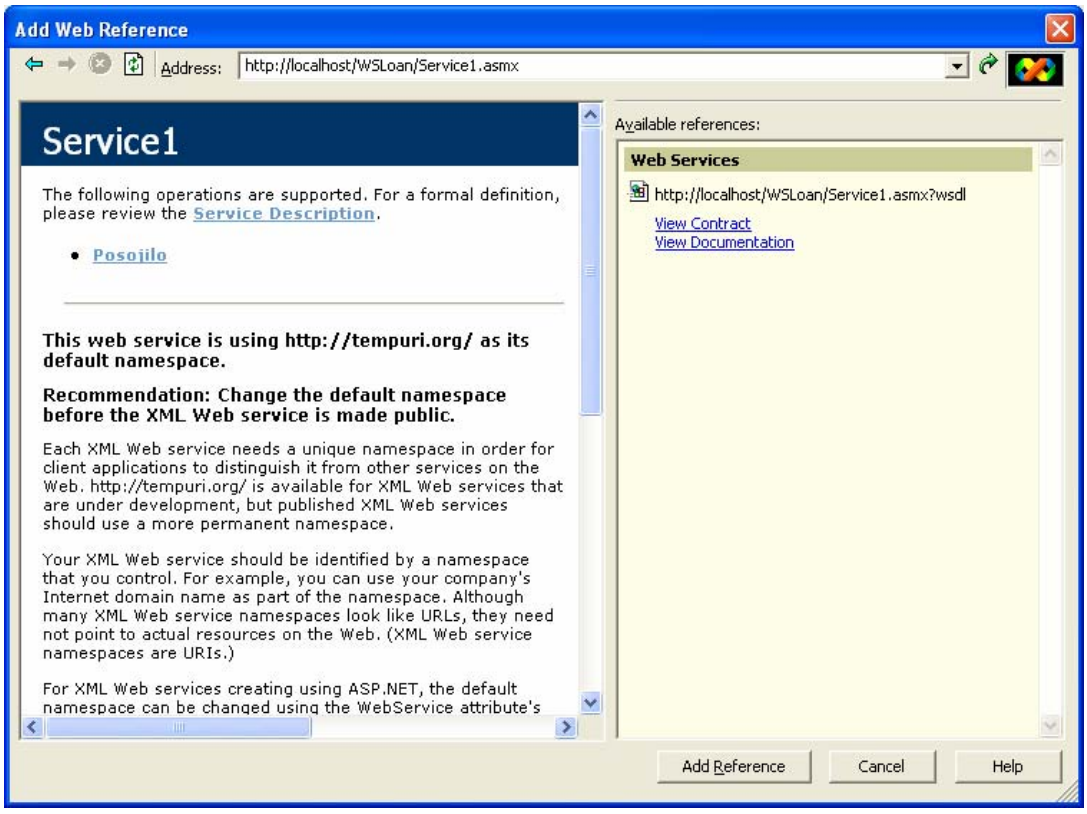
Spletno storitev sem do sedaj preizkusil znotraj internetnega raziskovalca. Kot primer si poglejmo, kako jo lahko uporabimo v povezavi z Windows aplikacijo. Za ta namen sem z razvojnim orodjem Visual Basic napravil Windows aplikacijo.

Slika 14: Windows obrazec, ki uporablja spletno storitev



Najprej je potrebno dodati referenco na prej ustvarjeno spletno storitev, da lahko dostopamo do njenih metod. To preprosto dosežemo tako, da v pogovorno okno vpišemo URL naslov, kjer se nahaja ».asmx« datoteka spletne storitve.

Slika 15: Dodajanje reference za spletno storitev



Potem, ko dodamo referenco se lahko v programski kodi sklicujemo na lastnost ali metodo enako kot na ostale lastnosti ali metode, ki so že vgrajene v orodju Visual Studio.

Rezultat, ki smo ga dobili s pomočjo spletne storitve lahko pošljemo v drugo aplikacijo ali zapišemo v datoteko, ki jo uporablja druga aplikacija. Tako lahko dodamo na primer proceduro, ki zapiše vrnjeni podatek o znesku posojila v lokalno datoteko z imenom »Rezultat.xml«.

Slika 16: Kako rezultat shranimo v lokalno datoteko

```
Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
localDS.WriteXML("c:\rezultat.xml")
End Sub
```

Slika 17: Uporaba spletne storitve za izračun obroka posojila (Windows aplikacija)

```
Public Class Form1
   Inherits System.Windows.Forms.Form

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Dim ws As New localhost.Service1()
      Dim Znesek, stMesecev, Obresti As Decimal
      Dim PlaciloZacMeseca As Boolean
      Try
         Znesek = CDec(txtZnesek.Text)
         stMesecev = CInt(txtstMesecev.Text)
         Obresti = CSng(txtObresti.Text)
         PlaciloZacMeseca = False
         If chkPlaciloZacMeseca.Checked Then PlaciloZacMeseca = True
      Catch DataException As SystemException
         MsgBox(DataException.ToString)
         Exit Sub
      End Try
      Try
         txtPayment.Text = -Math.Round(ws.Posojilo(Znesek,    stMesecev, Obresti,
PlaciloZacMeseca), 2)
      Catch calcException As Exception
         MsgBox(calcException.Message)
      End Try
   End Sub
End Class
```

Ugotovimo lahko, da se programiranje za spletno storitev ne razlikuje veliko od klasičnega programiranja. Rezultat dobimo v XML obliki, in ga lahko uporabimo v naši kodi, kot rezultat klicanja običajne funkcije. Kodi se doda pooblastilo do spletne storitve, zato lahko kličemo metode storitve, kot da bi klicali metode svojega razreda, razrede/metode ogrodja ali Windows API-je.

5. APLIKACIJSKA OGRODJA V PORAZDELJENIH SISTEMIH

S pojavom lokalnih računalniških mrež so se pojavile tendence po vzpostavitvi sistema, ki bi omogočal izkoriščanje večjega števila procesorjev, ki jih povezujejo hitra omrežja. Programska

oprema takega porazdeljenega sistema je bistveno drugačna. Sisteme lahko povežemo s t.i. mehko povezavo (loosely coupled), ki dopušča praktično avtonomno delovanje posameznih računalnikov. Vse bolj redko pa se uporabljajo tesno sklopljeni sistemi. Razvoj porazdeljeni sistemov poteka v smeri koncepta, ki uporabniku ustvarja iluzijo, da dela pravzaprav za enim sistemom.

Aplikacijsko ogrodje v porazdeljenih sistemih sestavljajo programske rešitve (API), ki omogočajo razvijalcu dostop do komponent, storitev, protokolov, itd. Zaželeno je, da ima aplikacijsko ogrodje naslednje lastnosti (Bajec, 2003):

- Podpora za več jezikov;
- Podpora za dostop do zastareli sistemov;
- Podpora za transakcijsko intenzivne sisteme;
- Podpora komunikaciji prek sporočil;
- Podpora programiranju spletnega strežnika;
- Podpora XML;
- Vmesniki do standardnih protokolov;
- Podpora povezavi z različnimi SUBP;
- Podpora dodeljevanju imen;
- Podpora varnosti.

5.1. Porazdeljeni objekti

Sodobni programski jeziki so objektni, ki omogočajo dedovanje, ograjevanje in enkapsulacijo. Ti elementi so se izkazali za odlične pri razvoju namiznih aplikacij. Problem nastopi, ko želimo zagotoviti povezovanje in sodelovanje različnih procesov, ker so vsi ti mehanizmi vezani na konkretne jezike. V tem primeru so se morali razvijalci zateči k nizkonivojskim rešitvam, kot so vtičnice (sockets), da so lahko gradili porazdeljene sisteme. Tehnologija porazdeljenih objektov je nadgradila objektno programiranje z možnostmi povezovanja objektov na različnih računalnikih. Za razvijalce sploh ni razlike med lokalnim in oddaljeni objektom, vsi so videti kot lokalni. Razvilo se je kar nekaj tehnologij porazdeljenih objektov, danes pa sta najpomembnejši CORBA in DCOM. Pomembno vlogo predstavlja tudi JavaBeans tehnologija, ki je vezana na programski jezik Java.

5.1.1. CORBA

CORBA natančno določa vmesnik za sodelovanje med različnimi objekti. Zastavljena je na tak način, da je dosežena neodvisnost od programskega jezika, v katerem razvijamo objekte. Prav tako se lahko objekti nahajajo na različnih platformah. Tako lahko razvijamo objekte v Javi, C/C++, Cobolu, itd. Dodatno CORBA nudi mehanizme, ki naredijo ovojnico okrog že obstoječih programov, za kar ne potrebujemo niti izvorne kode, s pomočjo katerih s temi programi delamo kot z objekti. Ker je CORBA na tržišču že precej časa (prva specifikacija je

bila razvita že pred eksplozijo interneta), je zelo zrela arhitektura. Nudi tudi precej storitev (imenovanje, varnost, spremljanje različic, itd.), ki so ključnega pomena za razvijalce aplikacij. Ker pa je v konzorciju, ki določa specifikacijo CORBA, združenih več kot sedemsto podjetij, je med njimi potrebnega precej usklajevanja, preden ponudijo novo različico specifikacij.

5.1.2. DCOM

DCOM (Distributed Component Object Model) je Microsoftova razširitev objektnega modela COM, prilagojena za porazdeljene aplikacije. Glavna prednost tega modela pred CORBA je v tem, da je podpora zanj vgrajena že v Microsoftov operacijski sistem, podprt pa je z mnogo več orodji za razvoj kot CORBA. Na drugi strani pa sta COM in DCOM omejena samo na Windows operacijski sistem, kar je dokaj slabo v primerjavi s CORBA, ki je podprta na različnih platformah.

Funkcionalnost DCOM modela je podobna funkcionalnosti CORBA modela. Razvijalci se pri razvoju strežniških aplikacij odločajo za CORBA, saj podpira precej različnih platform. Ko pa se razvija aplikacije na odjemalcih, se navadno uporabi COM/DCOM, ker so odjemalci praviloma Windows aplikacije.

5.1.3. JavaBeans

JavaBeans je model za razvoj komponent v programskem jeziku Java. Čeprav je razvoj komponent vezan samo na en programski jezik, je zaradi platformne prenosljivosti Jave zelo pomemben model. Prednost JavaBeans tehnologije je v tem, da je razvita na novo in zato ni omejena na predhodne tehnologije. JavaBeans ni produkt, program ali razvojno okolje. JavaBeans je dejansko Java razred, kateremu so dodane zunanje funkcionalnosti in specifikacije. Zaradi JavaBeans so bile potrebne majhne spremembe v Javi, ki so bile narejene v različici Java 1.1. JavaBeans spremenijo razrede v programske komponente, ki zagotavljajo več posebnosti. Nekatere posebnosti so značilne samo za JavaBeans. Programske komponente imajo lastnosti, metode in dogodke, ki so atributi objekta. JavaBeans so načrtovani tako, da lahko z lahkoto komunicirajo z obstoječimi komponentnimi tehnologijami.

5.2. .NET aplikacijska arhitektura

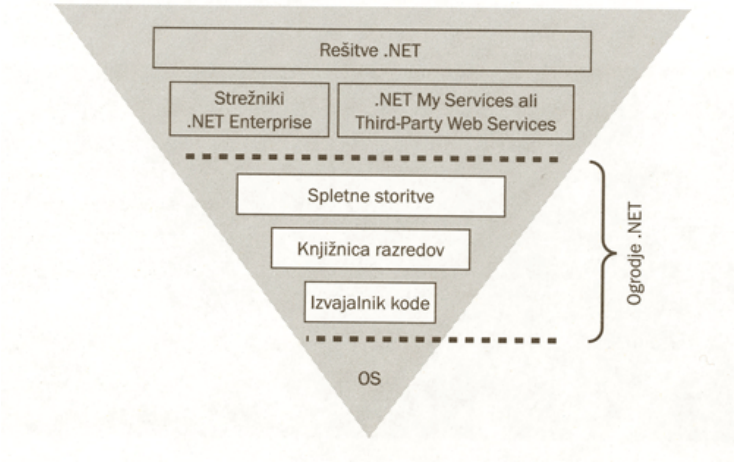
.NET nedvoumno predstavlja evolucijski korak, morda celo skok. Res je da .NET uporablja stare internetne tehnologije, ki so Oraclu in drugim računalniškim podjetjem prinesle velikanske dobičke ter pospešile ekonomski razvoj. .NET, kot že daje slutiti ime, je internetno usmerjen. Če izhajamo iz pred internetnih dni, to ni COM, ki so ga pakirali v ActiveX kot internetno rešitev. .NET ima sposobnost gladke integracije s spletom, ker je zgrajen z uporabo istih odprtih in splošno sprejetih standardov. Dobršen del knjižnice ogrodja .NET – podzbirke, neodvisne od operacijskega sistema – so podjetja Intel, Microsoft in Hewlett-Packard predala v

standardizacijo združenju ECMA. Tako lahko pričakujemo v bližnji prihodnosti tudi eno ali več različic, ki niso zasnovane na operacijskem sistemu Windows.

Ogrodje .NET predstavlja glavno komponento platforme .NET. Dejansko je njegova osnova in brez nje razvijalci ne morejo ustvarjati rešitev .NET. Obstajajo različne interpretacije o tem, kaj vse predstavlja ogrodje .NET. Težava je v subjektivnosti; če ne bo Microsoft jasno in konsistentno določil, kaj je ogrodje za .NET, se bo razprava nadaljevala v nedogled. Večina literature omenja, da sestavljajo ogrodje .NET naslednji tri deli:

- izvajalnik kode skupnega jezika (CLR – Common Language Runtime),
- knjižnice razredov,
- storitve: sistemske (znotraj .NET) in spletne storitve.

Slika 18: Pregled platforme .NET



Vir: Bart, 2002

5.2.1. Platforma .NET

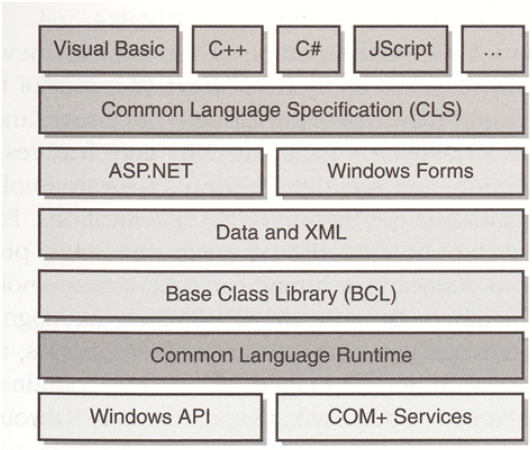
Komponente ogrodja omogočajo delovanje rešitev .NET. Microsoft je uporabil za pripravo platforme .NET enake razrede, kot so uporabljeni v knjižnici ogrodja. Z razvojnega vidika je to zelo pomembno. Za izdelavo rešitve torej uporabimo enake komponente iz katerih je zgrajena sama platforma. To nam zagotavlja, da bodo stvari tudi v resnici delovale, in ne samo, kako je v dokumentaciji zapisano. Komponente ogrodja prinašajo platformi veliko izbiro storitev in funkcij, med drugimi (Bart 2002, str. 52):

- Vhod/izhod (V/I). V/I vključuje operacije, kot so branje ali pisanje v datoteko, posredovanje tokov podatkov v različne smeri in iz njih, prikazovanje besedila na zaslon ali tiskalnik itd.
- Uporablja spletno tehnologijo in storitve, med drugim HTTP, HTML, XML in SOAP.
- Zagotavlja elemente uporabniškega vmesnika (UI) za izdelavo spletnih strani ali tradicionalne uporabniške vmesnike Windows, ki temeljijo na obrazcih.

- Poenostavlja izdelavo dinamičnih spletnih strani.
- Omogoča ločevanje aplikacijske logike in uporabniških vmesnikov pri izdelavi dinamičnih spletnih strani z uporabo ASP.NET.
- Poenostavlja delo s podatki iz različnih podatkovnih virov z uporabo ADO.NET.
- Zagotavlja varnost aplikacij in sistema.
- Uporablja izvajalne funkcije, npr. smetarja za izboljšanje stabilnosti aplikacij.
- Zagotavlja način za izdelavo in dostavo spletnih storitev.
- Omogoča souporabo ducatov razvojnih jezikov, tudi najnovejšega C#.
- Omogoča razvoj in izvajanje v različnih jezikih.
- Zavaruje vlaganja v obstoječe rešitve in hkrati omogoča bodočo razvojno pot.
- Ponuja eno ogrodje za izdelavo in dostavo aplikacij v katero koli napravo.
- Prinaša nove poslovne priložnosti zaradi novih vrst rešitev.

Kljub določenemu napredku v preteklih letih, je še vedno zelo zahtevno razviti robustne poslovne aplikacije. Za te namene potrebujemo zelo izkušene programerje. Programiranje je seveda zahtevno delo, vendar je veliko dodatnih dejavnikov, ki dodatno otežujejo programiranje. Windows platforma se je v preteklih letih razvijala kaotično. Dodatno težavo pri razvoju porazdeljenih sistemov predstavljajo omejitve objektov COM (Component Object Model). Na .NET lahko gledamo tudi kot naslednjo generacijo COM. V ozadju se popolnoma razlikujeta, in .NET je v primerjavi s COM boljši v mnogo pogledih (Balena 2002, str. 10). Za razliko od COM, .NET okvir uporablja koncept dedovanja. Vsi objekti v .NET okvirju so hierarhično razporejeni v osnovni imenik »System.Object class«, iz katerega so izpeljani vsi ostali razredi. Ti razredi oskrbijo skoraj vse funkcionalnosti, vključujoč uporabniški vmesnik, spletno programiranje, XML procesiranje, varnost in komunikacijo med računalniki. .NET okvir vsebuje več 3400 različnih razredov.

Slika 19: .NET arhitektura



Vir: Balena, Francesco, 2002

Windows API vsebuje sistemske funcije za Windows. Večina funkcij, ki jih kličemo v .NET okvirju, se dejansko nahajajo v Windows kernel DLL-ju. Začudimo se lahko, da imamo v arhitekturi tudi sloj COM+. Za razumevanje tega se je potrebno nekoliko ozreti v preteklost.

Izvajalnik kode -CLR (Common Language Runtime) je prvi sloj, ki pripada .NET okvirju. Ta sloj je zadolžen za osnovne storitve, kot so upravljanje s spominom in z izjemami ter večnitnost. Če bo Microsoft razvil .NET tudi za ostale operacijske sisteme, bo potrebno spremeniti CLR.

BCL (Base Class Library) predstavlja sloj, ki definira vse osnovne podatkovne tipe. BCL vsebuje tudi razrede za upravljanje .NET, kot so V/I datoteke in varnost. Podatkovne tipe določajo specifikacije CTS (Common Type system). Te specifikacije določajo, kako bo predstavljeno na primer polje s svojimi lastnostmi, dogodki; definirajo tudi kako se bo podatkovni tip dedoval od drugega tipa. Ker vsi jeziki v .NET prepoznajo te specifikacije, si lahko med seboj izmenjujejo podatke, kličejo v različne razrede med seboj in celo dedujejo razrede, ki so napisani v drugem jeziku.

Podatkovni in XML sloj vsebuje .NET razrede, ki delujejo s podatkovnimi bazami in XML-om. Podpora za XML je direktno vgrajena v .NET okvir, bolje kot z uporabo zunanjih komponent, kot je to bil primer s predhodniki .NET-a. V XML formatu ima .NET zapisane vse virtualne informacije. Vsa .NET konfiguracija je zapisana v XML obliki,in vsaki objekt se lahko shrani v XML obliki s samo nekaj ukazi. Seveda ima .NET okvir tudi dober in hiter XML urejevalnik. Podatkovni del tega sloja predstavlja ADO.NET. Kljub podobnemu imenu je med ADO in ADO.NET velika razlika. ADO.NET je osredotočen v glavnem na odklopljene podatkovne zapise – *Dataset* in ne podpira kazalnike na strani strežnika. *Dataset* objekt je zmogljivejši od *ADO Recorset* objekta, shrani lahko podatke iz več različnih podatkovnih baz v isto ali različne baze. Vzpostavimo lahko celo relacije med različnimi tabelami in uvozimo ali izvozimo meta podatke kot XML.

ASP.NET in Windows obrazci se nahajajo na istem nivoju. Ta del okvirja vsebuje vse razrede, ki generirajo uporabniški vmesnik. Kljub temu, da se ASP.NET in Windows obrazci nahajajo na istem nivoju, je velika razlika v tehnologiji med spletnim in Windows obrazcem. Spletni obrazec nastane na strežniku v HTML obliki; uporabnik ga pa vidi v internetnem brskalniku. Windows obrazci se kreirajo na strani uporabnika (uporabnik mora imeti Windows operacijski sistem). Lahko pa v isti aplikaciji uporabljamo oba tipa obrazcev.

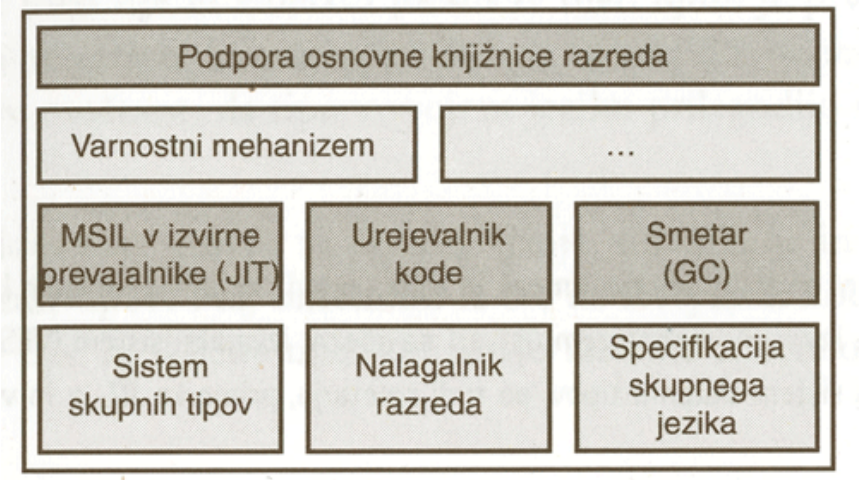
CLS (Common Language Specification) predstavlja seznam specifikacij. Te specifikacije diktirajo zahteve, ki jih mora imeti .NET jezik. Visual Basic se popolnoma ujema s specifikacijami CLS; pri C# temu ni tako, zato imamo možnost nastavitve v Visual Studio .NET s katero zagotovimo CLS kompatibilnost.

Na vrhu diagrama so jeziki, ki izpolnjujejo pogoje CLS-a. Zraven že omenjenih imamo še: COBOL, Pascal, SmallTalk, Eiffel, Python in APL. .NET v tem trenutku podpira sedemintrideset programskih jezikov. Zaradi tega, ker vsi .NET jeziki od Microsofta ali ostalih prodajalcev ciljajo na CLS in uporabljajo razrede ter podatkovne tipe iz okvirja .NET, se med seboj razlikujejo mnogo manj kot v preteklosti. Na primer CLR in CTS podpirata samo samostojno dedovanje (pomeni, da se tip lahko deduje samo od enega tipa), vsi jeziki podpirajo samostojno dedovanje; torej ni prostora za mnogokratno dedovanje v .NET-u. Iz enakih razlogov so tudi razlike v hitrosti med različnimi jeziki minimalne. Zato bi morala izbira jezika bazirati glede na izkušnje programerjev in so odvečne razprave o tem kateri jezik je boljši. Glede na to, da je najpomembnejše poznavanje .NET okvirja, se lahko s tem znanjem pozneje z lahkoto naučimo tudi drugi jezik.

5.2.2. Izvajalnik kode skupnega jezika - CLR (Common Language Runtime)

Poglejmo si izvajalnik kode skupnega jezika nekoliko natančneje. .NET platforma se po tem razlikuje od skoraj vseh drugih platform. Za razliko od Jave lahko izvajalnik kode .NET izvaja programe, napisane v katerem koli jeziku, ki izpolnjujejo specifikacijo skupnega jezika. Od tod tudi ime izvajalnik kode »skupnega jezika«. To pomeni, da lahko programi, napisani v enem jeziku, kličejo programe ali komponente v drugih jezikih.

Slika 20: Izvajalnik kode skupnega jezika



Vir: Bart, 2002

5.2.3. Sistem skupnih tipov

Sistem skupnih tipov predstavlja središče izvajalnika kode. S sistemom je določena skupina pravil in »lastnosti« , ki so skupne večini jezikov in s katerimi je mogoče različne jezike preslikati v platformo .NET. Sistem skupnih tipov ni izvajalni sistem, ampak je zbirka pravil,

ki jih mehanizem izvajalnika kode upošteva med izvajanjem kode. Pomembnejše lastnosti sistema skupnih tipov so (Bart, 2002 str. 85):

- Pravila za napoved, uporabo in upravljanje tipov.
- Pravila za uvajanje predmetno usmerjenega modela, v katerem je mogoče uporabiti večino jezikov.
- Pravila za obravnavo izjem.
- Napovedovanje podpore za dve kategoriji tipov (vrednostne in sklicne tipe).
- Dodatna pravila in zakonitosti za medjezikovno integracijo.

5.2.4. Smetar (Garbage collector)

Ogrodje .NET uporablja izpopolnjene tehnike upravljanja pomnilnika tako za dodeljevanje kot za sproščanje pomnilnika med delovanjem aplikacije. Zakaj je tako pomembno upravljati pomnilnik? Običajno se pomnilnik programu dodeli na nekem mestu, nato pa posreduje naprej in je uporabljen na različnih krajih. Nekje na tej poti se pomnilnik sprosti (vrne operacijskemu sistemu). Če pa kateri del istega programa ohranja sklic do tega sproščenega pomnilnika (ni seznanjen z dejstvom, da je bil pomnilnik sproščen) in ga skuša vseeno uporabiti, poten nastanejo težave (kršitev dostopa). Nasprotje tega je znano kot »puščanje pomnilnika«, ki se zgodi, če uporabnik pozabi sprostiti pomnilnik, vendar sprosti (ali nikoli več ne uporabi) sklic do pomnilnika. Tako je pozabljeni pomnilnik – kot vedro, ki po kapljicah pušča, dokler ni prazno. To traja tako dolgo dokler ne zmanjka pomnilnika. Puščanje pomnilnika je zelo težko odkriti. Omenjena težave ne predstavlja posebnega problema za namizne računalnike, ki ga lahko ponovno zaženemo in s tem vrnemo pomnilnik. Veliki problem pa lahko to predstavlja za strežniško programsko opremo. Strežniki so pogosto vklopljeni več mesecev ali celo nekaj let. Pomanjkanje pomnilnika pa slabo vpliva na stabilnost in varnost sistemov.

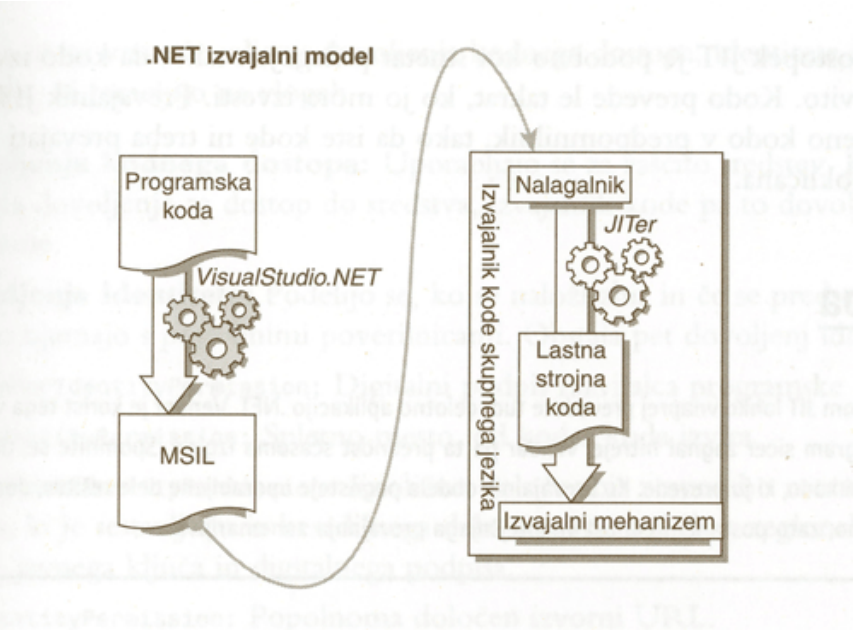
Programer sicer še vedno zahteva dodelitev pomnilnika, za sproščanje pomnilnika pa izvajalni program uporablja smetarja. Ker sistem prevzame odgovornost za obnavljanje neuporabljenega pomnilnika, so programi stabilnejši in varnejši. Ker smetar jamči, kdaj bo vrnil neuporabljeni pomnilnik, programerji ne morejo predvideti in naj ne bi predvidevali, v kakšnem stanju je pomnilnik, kar je običajno priporočljiv programerski postopek. .NET želi postati razvojna in izvajalska platforma prihodnosti za velika podjetja, kjer je ključnega pomena stabilnost. Ena izmed rešitev je, da se upravljanje pomnilnika vzame iz rok razvijalcev.

5.2.5. Prevajanje JIT (Just In Time)

Programi .NET se ne prevedejo neposredno v izvedljivo kodo, ampak v vmesni jezik t.i. MSIL ali IL (Microsoft Intermediary Language). Čeprav je izraz JIT poznan poznavalcem Jave, je potrebno poudariti, da ni nobene podobnosti z Javo. Za boljše razumevanje JIT-a si poglejmo najprej primer programske rešitve, ki ni napisana za .NET. Če razvijemo rešitev v C++, je rezultat strojna koda, ki je namenjena ciljnemu procesorju in je neposredno izvedljiva.

V .NET poteka vse to drugače. Ker .NET teži k optimizaciji in možnosti delovanja v katerikoli napravi, prevajalnik odda program v vmesnem jeziku MSIL, ki je v osnovi procesorsko neodvisen program in ga samega ni mogoče izvajati. Ko program zaženemo, izvajalnik kode skupnega jezika naloži kodo v izvajalno okolje, izvajalnik JIT pa IL prevede v sistemu lastno izvedljivo kodo. To lastno izvedljivo kodo nato izvede izvajalni mehanizem izvajalnika kode.

Slika 21: .NET izvajalni model



Vir: Bart, 2002

5.2.6. ASP

ASP (Active Server Pages) tehnologija je relativno mlada, predstavljena leta 1996. Pred to tehnologijo so razvijalci uporabljali za razvoj aktivnih internetnih strani CGI (Common Gateway Interface) in ISAPI[15] (Internet Server Application Programming Interface). CGI je bila prva tehnologija, ki se je množično uporabljala za razvoj aktivnih internetnih strani. CGI dinamično generira HTTP odgovor. Ponavadi se uporablja programski jezik C ali skriptni jezik, kot je na primer Perl. Spletna stran je lahko prilagojena uporabniku in zgrajena iz informacij, ki so shranjene v podatkovni bazi. Kljub temu ima CGI določene omejitve. Za vsako poizvedbo se kreira novi proces. Potem, ko je proces obravnavan, se mora končati pred pričetkom obdelave novega. Pri velikem številu zahtev potrebujemo zato zelo veliko procesorsko moč.

Microsoft je z razvojem ASP tehnologije odpravil prej omenjene omejitve CGI programiranja. ASP omogoča izvajanje kode znotraj spletne strani, kar občutno poenostavi spletno

[15] ISAPI se za razliko od CGI implementira kot izvršilna datoteka (exe) na Windows platformi. Ekstenzija je implementirana kot DLL (Dynamic Link Libraries), kar pomeni, da se naloži v memorijo samo ob prvi zahtevi.

programiranje aktivnih strani. Po letu 1996, je Microsoft izdal več različic ASP-ja. V letu 1998 je bila izdana različica ASP 2.0 in spletni strežnik IIS 4.0. V letu 2000 pa različica ASP 3.0 in ISS 5.0, ki je imela veliko boljše zmogljivosti od predhodnih (ob predpostavki, da smo uporabili Windows 2000). Kljub temu ima ASP naslednje pomanjkljivosti (Jason Butler, 2002) :

- ASP koda postane hitro nepregledna, ker ni ločena od ostalih podatkov spletne strani. Na isti spletni strani je pomešana koda, ki se izvaja na strani strežnika in koda, ki se izvaja na strani uporabnika. Zaradi tega je zelo težavno vzdrževati in spreminjati programsko kodo.
- ASP ne predstavlja pravega komponentnega modela. Programerji pričnejo s pisanjem kode na vrhu spletne strani in se premikajo proti dnu s pisanjem kode za poizvedbe v podatkovnih bazah, poganjanje poslovne logike in generiranje HTML vsebine.
- Koda je prepletena s predstavitvijo spletne strani. To predstavlja problem, kadar delajo na projektu skupaj programerji in oblikovalci spletne strani. Podpora za internacionalizacijo je otežena.
- Kombinacija ASP in ISS ni vedno najbolj zanesljiva platforma. Microsoft je naredil ASP platformo tako odprto, da je mogoče hitro narediti aplikacijo, pred katero odpove IIS. Razviti ASP aplikacijo je ena stvar, razviti dobro, učinkovito in zanesljivo aplikacijo pa je povsem druga stvar.
- Namestitev ASP aplikacije, ki vsebuje COM je lahko zelo težavno. COM objekti morajo biti registrirani, in so zaklenjeni s strani operacijskega sistema Windows, ko so v uporabi. Rezultat tega je, da je zelo težko uporabiti ASP aplikacijo, ki teče na več spletnih strežnikih.

5.2.7. ASP.NET

ASP.NET ne predstavlja samo izboljšano različico predhodnika, temveč evolucijski korak k razvoju spletno zasnovanih aplikacij. Gre za izboljšano tehnologijo ASP-ja ki predstavlja nov način za pripravo spletnih rešitev z uporabo spletnih obrazcev Web Forms. Kombinacija Windows Forms in Web Forms nam ponuja isti razvojni koncept in tehniko za izgradnjo interaktivnih aplikacij znotraj .NET – Form. Prednost je v tem, da lahko znanje iz Windows programiranja uporabimo tudi v spletnem programiranju. Čeprav zahtevajo te rešitve različne razrede, pa so si med seboj po konceptu zelo podobne. Najbolj značilna sprememba glede na ASP je tehnika, imenovana »Codebehind«, ki omogoča, da svoj uporabniški vmesnik ločimo od predstavitvene kode (HTML). Ta delitev predstavitvene in poslovne logike ima številne prednosti. Če načrtujemo spletni vmesnik, velikokrat ne načrtujemo poslovne logike. Ker sta vmesnik in logika ločeni datoteki, ju lahko različna uporabnika hkrati uporabljata. Prav tako pa je tveganje, da načrtovalec grafike poškoduje kodo za logiko aplikacije in tudi obratno bistveno manjše.

5.3. J2EE aplikacijska arhitektura

J2EE bazira na konceptu n-slojne arhitekture. Prednost te arhitekture je v ločitvi poslovne logike od podatkovnih zbirk ali infrastrukture. Za večnivojsko arhitekturo so značilni porazdeljeni sistemi. Najbolj znani sta dvo in tro-nivojska arhitektura. Dvo-nivojsko arhitekturo sestavljata:

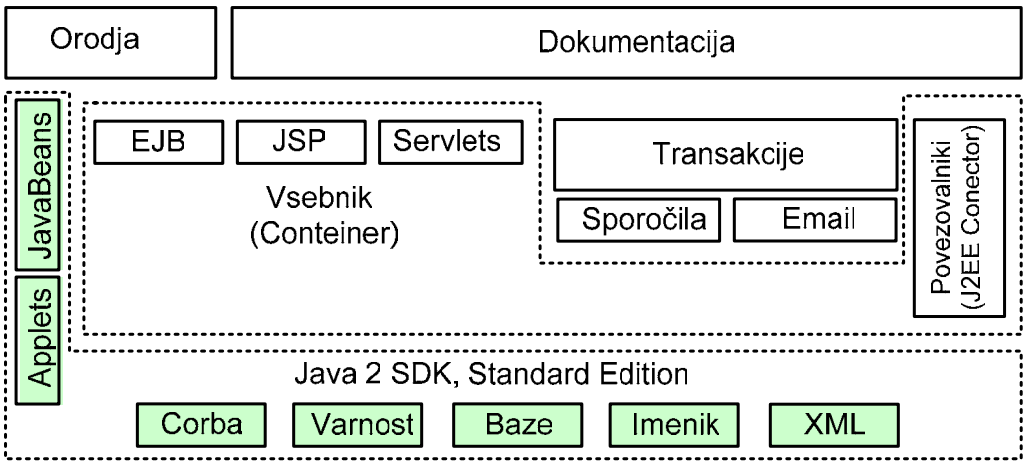
- **predstavitvena in logična raven** (predstavitev in procesiranje podatkov, ki jih vrne strežnik).
- **Podatkovna raven** (shramba podatkov), ki je fizično realizirana na strežniku.

Dvo-nivojska arhitektura je primerna tam, kjer je malo procesiranja podatkov (npr. spletni strežnik). Slabost dvo-nivojske arhitekture se pokaže pri velikem številu odjemalcev in veliki količini procesiranja podatkov. Posledica tega so veliki stroški skrbništva, delovanja in vzdrževanja sistema.

V poslovnih aplikacijah so ponavadi naslednji sloji:

- **Uporabniški sloj** predstavlja uporabniški vmesnik. Uporabnika lahko predstavlja namizna računalnik, internetni brskalnik, mobilni telefon ali PDA.
- **Predstavitveni sloj** je odgovoren za sprejemanje zahtev od uporabnika, interpretacijo teh zahtev in posredovanje zahtev poslovnemu sloju za procesiranje. Prezentacijski sloj nato zapakira odgovor in ga pošlje uporabniku.
- **Poslovni sloj** je središče vsake poslovne aplikacije. Vsebuje vso poslovno logiko.
- **Integracijski sloj** vsebuje programske module, ki so povezani na zunanje vire. Zunanji viri so lahko podatkovne baze, spletne storitve, datotečni sistemi itd.
- **Sloj virov** vsebuje podatke, ki so lahko zapisani v podatkovnih zbirkah ali drugih poslovnih sistemih (ERP, CRM, spletne storitve).

Slika 22: J2EE okvir



Vir: Jurič, 2001

J2EE uporablja za dostop do oddaljenih objektov predvsem tehnologijo CORBA, ki se lahko uporablja za različne programske jezike. Za podporo transakcijam je na voljo EJB (Enterprise Java Beans), ki predstavlja specifikacijo API za razvoj razširljivih, porazdeljenih, komponentnih in več-nivojskih aplikacij. EJB nam znatno olajšajo delo s transakcijami; možno je enostavno prenašati podatke iz enega transakcijskega strežnika na drugega. Za sprejemanje in oddajanje sporočil v porazdeljenih sistemih se uporablja Java Message Service. Pošiljanje sporočil zahteva definicijo protokola za komunikacijo. Protokol definira sporočila (ukaze), ki jih lahko odjemalec pošlje strežniku in obratno. Obstajajo stalni in prilagodljivi protokoli. Pošiljanje sporočil lahko poteka sinhrono ali asinhrono.

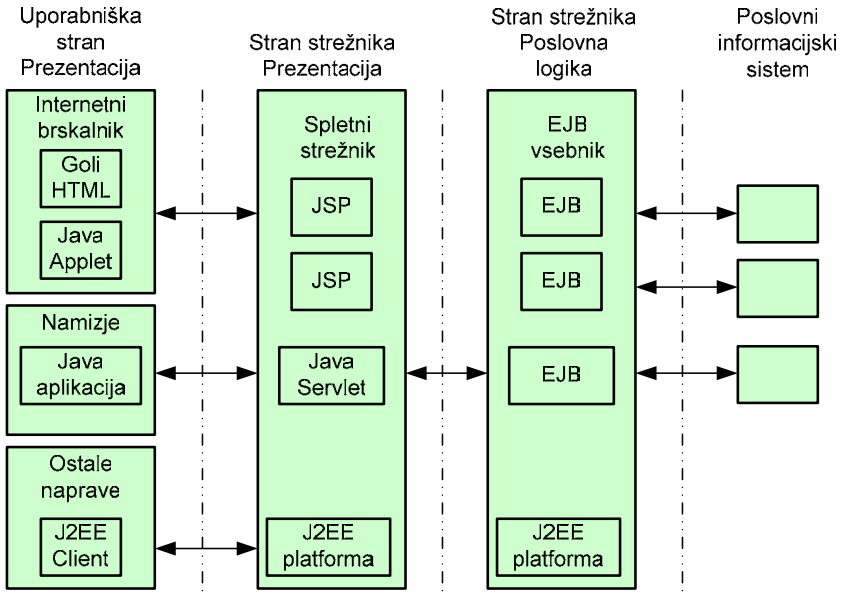
Imenik se uporablja za podporo dodeljevanja imen. Na voljo JNDI (Java Naming and Directory Interface), ki predstavlja API za dostop do obstoječih storitev (npr. LDAP strežnik).

Za dostop do zastarelih sistemov se uporablja J2EE Connector (podobno kot vmesnik JDBC – Java Database Connectivity).

J2EE vsebuje veliko število API-jev za interpretacijo XML-a. Za programiranje spletnega strežnika se najpogosteje uporabljajo Servleti in JSP (Java Server pages).

Za zagotavljanje varnosti se uporablja kriptografija. J2EE ima na voljo tudi razširitev JCE (Java Cryptography Extension), ki omogoča uporabo številnih varnostnih mehanizmov (ključi, digitalna potrdila, digitalno podpisovanje, itd.)

Slika 23:Vsebnik J2EE okvirja



Vir: Jurič, 2001

5.3.1. Appleti

Applet predstavlja osnovni mehanizem jezika Java za programiranje odjemalca. Spletna stran, ki je sprejeta od strežnika lahko vsebuje programski dodatek - applet. Appleta ni mogoče zagnati neposredno iz operacijskega sistema. Program se požene, ko internetni brskalnik zazna spletno stran z vključenim appletom. Za izvajanje appletov mora biti pri odjemalcu naloženi javanski navidezni stroj JVM. Appleti so v osnovni obliki zelo omejeni. Ne omogočajo dostopa do lokalnih datotek, poganjanja drugih programov, brisanje lokalnih datotek in povezovanja na druge računalnike. Razlog za te omejitve je v varnosti.

5.3.2. Poslovna logika na strežniku

J2EE komponente na strežniku so servleti ali JSP. Servleti so del kode v Javi, ki se naloži na spletni strežnik in izvede kot reakcija na nek HTTP ukaz. Servlete je mogoče uporabiti na večini spletnih strežnikov. Servleti so naloženi v pomnilniku, kar pomeni da je izvajanje hitrejše, ker ni potrebno vsakokrat na novo kreirati proces. Pomemben vidik servletov je zmožnost, da se podatki shranijo v spremenljivko in kasneje uporabijo, kar je temeljna težava začetne oblike spletnih strani. Servleti omogočajo dokaj visoko raven varnosti. Kodo servleta je potrebno najprej prevesti, nato pa je potrebno prevedeno datoteko prenesti na področje, ki je znotraj strežnika deklarirano za servlete. Servlet je potem dostopen prek URL naslova.

JSP predstavlja tehnologijo dinamičnih spletnih strani podobno Microsoftovem ASP-ju. Gre za popularen način programiranja spletnih strani na strani strežnika. JSP razvijalcu omogoča, da želeno funkcionalnost spletnih strani zapiše v Javi, kodo pa vključi v HTML stran. Za razliko od appletov, se JSP koda izvaja na strani strežnika. Tehnologija JSP je tesno povezana s servleti. Posebna komponenta strežnika obdela zapis v JSP formatu in ga pretvori v servlet.

5.3.3. Poslovne komponente

Poslovna logika, ki predstavlja logiko za potrebe določene poslovne domene kot so bančništvo, prodaja, finance, itd., se obravnava s poslovnimi zrni (bean) ki se nahajajo v poslovnem sloju. Poslovno zrno sprejme podatke od odjemalca, jih nato procesira (če je potrebno), in pošlje podatke v poslovni informacijski sistem v shrambo. Poslovno zrno lahko tudi sprejme podatke iz shrambe, jih nato procesira (če je potrebno), in jih nato pošlje odjemalcu. Obstajajo tri vrste poslovnih zrn: zrna seje, zrna entitete in sporočilna zrna. Sejna zrna predstavljajo začasno konverzacijo z odjemalcem. Ko odjemalec konča izvajanje, se podatki, ki jih vsebuje sejno zrno zgubijo. V nasprotju, entitetno zrno ohrani trajno podatke v eni vrstici podatkovne zbirke. Če odjemalec prekine povezavo ali če strežnik ugasnemo, podložena storitev poskrbi, da so podatki entitetnega zrna shranjeni. Sporočilno zrno predstavlja kombinacijo sejnega zrna in JMS (Java Message Service), ki dovoljuje poslovnim komponentam sprejemanje JMS sporočila asinhrono.

5.3.4. Poslovni informacijski sistem

Poslovni informacijski sistem vsebuje poslovne infrastrukturne sisteme kot so ERP, »mainframe« transakcijska procesiranja, sisteme za upravljanje baz podatkov (SUBP) in ostale stare poslovne aplikacije. Na primer, J2EE aplikacijske komponente mogoče potrebujejo dostop do poslovnega informacijskega sistema zaradi povezovanja s podatkovno bazo.

J2EE ima standardni protokol JDBC za dostop do številnih standardnih relacijskih podatkovnih baz. JDBC omogoča:

- Povezavo z bazo na lokalnem ali oddaljenem računalniku;
- Pošiljanje poizvedb (SQL stavki);
- Procesiranje rezultatov poizvedb.

Dostop do zastarelih aplikacij nam omogoča JCA (Java Connector Arhitecture), ki se uporablja podobno kot vmesnik JDBC. Za sprejemanje in pošiljanje sporočil v porazdeljenih sistemih se uporablja JMS (Java Message Service). Izmenjava informacij poteka asinhrono.

6. PRIMERJAVA J2EE IN .NET

Cilj raziskave je primerjava tehnologije J2EE in .NET. V člankih je veliko debat o tem, da je ena tehnologija mnogo boljša od druge. Beremo lahko naslove ».NET prehitel J2EE za 432%«. V nekem drugem članku bi lahko prebrali ravno nasprotno. Bolj resne raziskave kažejo na to, da sta obe tehnologiji po zmogljivosti približno enaki. Za poslovne aplikacije je zmogljivost pomembna ne pa tudi najpomembnejša. Z vidika poslovnih aplikacij bom v nadaljevanju naredil primerjavo teh dveh tehnologij glede na naslednja sodila:

- Odvisnost tehnologije od operacijskega sistema.
- Primerjava orodij, ki so na voljo za razvijalce programske opreme.
- Potrebna strojna oprema in stroški implementacije.
- Prenosljivost obstoječih aplikacij na novo razvojno okolje.
- Varnost in omejitev dostopa do kode.
- Analiza arhitekture in zrelost platforme.
- Odvisnost od programskega jezika
- Spletne storitve in možnosti integracije.
- Podpora prodajalcev in vezanost na enega prodajalca.

6.1. Upravljavski vidik

Svet se zelo hitro spreminja. Z razvojem interneta so se v zadnjih desetih letih skokovito povečale zahteve poslovnih aplikacij. Danes, poslovni partnerji, stranke in celo zaposleni pričakujejo direktni dostop do informacij o podjetju in njihovih produktih. Tako kot se spreminja svet, se mora temu primerno prilagajati tudi informacijska tehnologija v podjetju.

Preden se odločimo za določene spremembe v obstoječi informacijski tehnologiji, je potrebno razmisliti o naslednjem:

- Katera tehnologija je primerna za obstoječe okolje v podjetju.
- Kako izbrati primernega proizvajalca programske opreme.
- Je naša organizacija homogena ali heterogena.
- Ali poznamo izkušnje tehničnega kadra v podjetju.
- Ali načeloma kupujemo ali razvijamo programske rešitve.
- Katera tehnologija ponuja več novosti: J2EE ali Microsoft .NET.

Razvijalci informacijskih tehnologij so v zadnjih letih trdo delali na tem, da najdejo rešitve za nove potrebe poslovnih aplikacij. Rezultat tega sta dva glavna aplikacijska razvijalska modela: Java 2 Enterprise Edition (J2EE) od podjetja Sun Microsystems[16] in Microsoft .NET. Čeprav imata ta dva modela podobno arhitekturo, se precej razlikujeta v marketinški usmeritvi. J2EE sloni na odprti kodi in skupnosti, ki usmerja razvoj te kode. .NET je usmerjen na Windows 2000/XP platformo. Prav zaradi te razlike poteka mnogo neskončnih diskusij.

Preden se odločimo, katero arhitekturo bomo uporabljali, je potrebno skrbno analizirati vse storitve, ki jih ponuja J2EE skupnost in Microsoft.

6.1.1. Omejitve zaradi operacijskega sistema

Problem aplikacij današnjega časa je pogosto v tem, da so povezane ali omejene na določeni operacijski sistem. Če imamo npr. rešitev pripravljeno za Windows, ta ne bo delovala niti v operacijskem sistemu Linux, OS/2, MacOS niti v katerem drugem sistemu. Za povprečnega uporabnika to ne predstavlja posebne težave. Nasprotno, uporabnik v podjetju uporablja različne operacijske sisteme: od sistemov Microsoft Windows do IBM AIX in vsega kar je vmes, vključno s trendovskim odprtim operacijskim sistemom Linux. V tako raznolikem okolju utegne biti uvajanje rešitve, ki je omejena le na en operacijski sistem, omejevalno ali celo nespremenljivo.

[16] Sun Microsystems je ameriško podjetje, ustanovljeno leta 1982. Njihova vizija je: vsak in vsi morajo biti priključeni na omrežje

Večina težav je zakoreninjena v arhitekturi operacijskih sistemov. Včasih se organizacije skušajo lotiti problema z vprašanjem »Katera je ciljna platforma?«. Odgovor je lahko katerikoli operacijski sistem, ki najbolj ustreza rešitvi, ki odpravi uporabnikove težave in pri tem ni pomembno, ali uporablja najbolj učinkovit operacijski sistem. Problem je v tem, ker je na voljo mnogo različnih operacijskih sistemov, a le malo orodij, ki ponujajo eno rešitev za več sistemov, ki jih lahko uporabljajo potencialni kupci. Razlog, da je tako malo orodij in metod, ki bi delovale z več jeziki in operacijskimi sistemi, je v tem, da jih ni preprosto ustvariti.

Operacijski sistemi omejujejo tudi izbiro programskega jezika. Idealno bi bilo, če bi lahko razvojna skupina izbrala najboljši programski jezik. Tako bi npr. pri izdelavi internetne aplikacije, ki je tipa odjemalec/strežnik, izbrala Javo zaradi prenosljivosti in preprostega razvijanja, na odjemalski strani pa bi izbrala Visula Basic 6. Ovira je v tem, da Visual Basic 6 deluje le v sistemu Windows. To pomeni, da mora razvojna skupina za vse odjemalske aplikacije uporabiti drugačno rešitev – tako, ki je npr. primerna za Solaris in Windows – ali pa uporabiti dve orodji, se pravi izdelati dve odjemalski rešitvi.

Tudi standardizirani programski jeziki ne rešujejo zgoraj omenjenih težav. Npr. C++ je standardiziral ameriški inštitut za standarde (ANSI) in mednarodni inštitut za standarde (ISO). Standard določa slovnico in sintakso jezika ter vsebuje informacije, ki jih proizvajalci orodij potrebujejo za vpeljavo jezikovnih prevajalnikov. Tako imajo proizvajalci programske opreme na voljo vse potrebne podrobnosti za uvajanje jezikov v različnih operacijskih sistemih. To pomeni, da lahko naredijo razvojna orodja, ki prevajajo programe C++ v katerem koli operacijskem sistemu. Vendar je kljub temu na tržišču le malo prevajalnikov namenjenim več operacijskim sistemom.

6.1.2. Osnovne definicije

Obe tehnologiji .NET IN J2EE sta sestavljeni iz ogrodja (framework), v katerem imamo različna programska orodja za razvijanje in upravljanje aplikacij. V obeh primerih se predpostavlja, da bodo aplikacije delovale v različnih okoljih – od strežnikov, osebnih računalnikov in prenosnih naprav.

J2EE je v osnovi niz standardov. Ko primerjamo J2EE z ostalimi tehnologijami, primerjamo dejansko veliko število nevtralnih platform, ki podpirajo standard J2EE, na osnovi katerih so razvite Java aplikacije. Več kot petdeset prodajalcev je razvilo orodja, strežnike itd. za pospešeno razvijanje tisoče razredov, podatkovnih struktur in podatkovnih tipov. Ti prodajalci, vključujoč IBM, BEA, Sun in ostali, so poenostavili razvijanje aplikacij z zagotovitvijo poenostavljenih podatkovnih dostopov skozi JDBC, čisto integracijo v obstoječe sisteme z JCA (J2EE Connector Arhitecture) in široko podporo za XML in spletne storitve. Ker je tako velika skupnost vpletena v razvoj in podporo programske opreme za J2EE specifikacijo, je med njimi izredno velika tekmovalnost in konkurenca. Zaradi tega so podjetja pričela razvijati svoje

specifikacije, vse z željo, da bi dosegla najboljši produkt za primerno ceno. Tako lahko arhitekti izbirajo svoje individualne komponente in jih vključujejo v svoje programske rešitve.

Na drugi strani imamo .NET produkt, ki ga sestavlja veliko število novih tehnologij. .NET je v osnovi načrtovan za Windows platformo. Microsoft promovira .NET kot popolnoma novo platformo, posebej načrtovano za internet. Aplikacije v .NET lahko razvijamo z različnimi programskimi jeziki, kot so: C#, VB.NET, J# in Cobol. Skupno število programskih jezikov, ki jih podpira .NET je trenutno 37. .NET si lahko predstavljamo kot tehnologijo, ki uporablja platformo Windows, nevtralna pa je glede izbire programskega jezika.

Do sedaj je Microsoft vključil XML podporo v skoraj vse programe, ki so namenjeni za komunikacijo in shranjevanje podatkov (»XML to the core«). Dostop do podatkovnih baz je poenostavljen s pomočjo na novo zgrajenega podatkovnega mehanizma ADO.NET. Pred leti je bila Windows platforma precej kaotična. Obstajali so najmanj trije različni programski modeli za izdelavo grafično intenzivnih aplikacij (GDI, DirectX in OpenGL). Vsi ti modeli so se med seboj razlikovali. Microsoft je proizvedel tudi več programskih modelov za dostopanje do podatkovnih baz – DAO (Data Access Objects), RDO (Remote Data Objects) in ADO (ActiveX Data Objects), ki so bili med seboj nezdružljivi.

Z razvojem ADO.NET se je bistveno izboljšala platforma internetnega programiranja. .NET nam sedaj omogoča razvijanje individualnih komponent z uporabo različnih programskih jezikov. Sestavlja ga približno 3400 razredov, podatkovnih tipov in struktur. Vse to omogoča CLR (Common Language Runtime), ki predstavlja podobno okolje kot JVM (Java Virtual Machine).

J2EE bazira na Javi, ki predstavlja sodoben objektno orientiran jezik. Java se nenehno spreminja in predstavlja pravo revolucijo na področju razvoja programske opreme. V Javi so osnova objekti, ki predstavljajo programske enote, sestavljene iz spremenljivk in z njimi povezanih metod. Ena izmed najmočnejših lastnosti Jave je dedovanje razredov. To je mehanizem, pri katerem razredi podedujejo svoje stanje v obliki deklaracij spremenljivk in vedenje v obliki metod od svojih razredov. Na ta način lahko razvijalec v izpeljanih razredih kodo ponovno uporablja in naredi svoje metode. Posebnost javanskih aplikacij je, da so platformsko neodvisne. Tako lahko isto kodo uporabljamo na platformah Windowsa, Unixa ali Linuxa. Da pa lahko Javo poganjamo na določeni strojni opremi, potrebujemo javanski navidezni stroj JVM (Java Virtual Machine), ki ukaze napisane v javanski kodi, pretvori v binarno obliko, ki jo procesor zna izvesti. Vsaka nova verzija JVM izboljša hitrost javanske kode.

Za Javo je značilna visoka zanesljivost in nizki stroški vzdrževanja. Ocenjuje se, da v Javi razvija aplikacije več kot 2,5 milijonov profesionalnih programerjev, ki so v zadnjih petih letih

razvili J2EE bazirane produkte. Ta kolektivna izkušnja je veliki prispevek k izboljšanju zanesljivosti in zmogljivosti J2EE. .NET se je osredotočil na izboljšanje zanesljivosti in zmogljivosti. Visual Studio .NET zagotavlja koherentno platformo za zmanjšanje ali odpravo hroščev v programski kodi. .NET vključuje tudi izboljšano varnost dostopa do programske kode.

6.1.3. Orodja

Za razvoj Java aplikacij imamo na razpolago zelo veliko število programskih orodij. Najbolj poznana so naslednja: JavaAge for Java, Visual Cafe, JBuilder in veliko orodij, ki temeljijo na odprti kodi. Glede na tako veliko število orodij, je očitno, da so ta orodja prilagojena za posebne namene in nudijo bistveno več funkcionalnosti kot .NET razvijalska skupnost, ki jo podpira Microsoft. Potrebno je upoštevati, da za doseganje vseh potrebnih funkcionalnosti v J2EE potrebujemo veliko število orodij in veliko teh orodij med seboj ni kompatibilnih. Kakorkoli, Sun, Oracle in Borland delajo veliko na tem, da združijo ta orodja v novi produkt in poenostavijo razvijanje aplikacij v J2EE.

Osnovni IDE za razvijanje aplikacij v .NET je Microsoftov Visual Studio .NET. Visual Studio ima integrirana vsa potrebna orodja, ki so, tako kot je za Microsoftove izdelke značilno, uporabniško zelo prijazna. Na voljo je veliko čarovnikov, ki lahko občutno poenostavijo in pohitrijo določena opravila. Vsebuje projektne predloge za ustvarjanje skoraj vseh vrst rešitev .NET, ki si jih lahko zamislimo, med drugim tudi aplikacij z obrazci Windows – kar so pravzaprav tradicionalne (Microsoft Windows) aplikacije uporabniškega vmesnika – spletnih storitev, aplikacij ASP.NET, rešitev podatkovnih zbirk in rešitev za podjetja. Visual Studio je boljši od predhodnika (Visual Basic 6.0) v vizualnem programiranju. Združuje moč iz Microsoft Visual C++ ter preprostost programskega modela obrazcev in razvojno okolje iz Visual Basica 6.0. Večji del aplikacije je mogoče izdelati z uporabo vzorca povleci-in-spusti. Večjo učinkovitost razvijanja omogoča tudi podpora za podatkovne zbirke in čarovniki, ki pomagajo povezati rešitev s podatkovno shrambo. V Visual Studio je vključena tudi podpora za HTML, XML, XSLT. Razvijalcem je v veliko pomoč tudi tako imenovana »dinamična pomoč«, ki aktivno odpira teme pomoči v zvezi s trenutno vsebino kode, s katero trenutno delamo.

Strojna oprema in stroški implementacije

.NET je optimiziran za Windows okolje. Ko gledamo na zanesljivost, zagotavljata J2EE in .NET zelo visoko zanesljivost, vendar na različni način. .NET si z lahkoto izposoja procesorsko moč v tako imenovani obliki »grozdenja strežnikov«, medtem ko J2EE doseže tradicionalno »konjsko moč« z izkoriščanjem več in hitrejših procesorjev. .NET implementacija je ponavadi bolj primerna za aplikacije nizkega cenovnega razreda. Za implementacijo zahtevnejših programskih rešitev nam omogoča implementacija z J2EE nižje stroške.

6.1.4. Preselitev aplikacije

Preden se podjetje odloči, katero tehnologijo bo uporabilo, mora upoštevati tudi obstoječe aplikacije. Pri preselitvi aplikacij so možni naslednji scenariji:

- Preselitev obstoječe Java aplikacije v novo verzijo Java aplikacije.
- J2EE aplikacije so po svoji naravi že prenosljive. Tradicionalno, Java teče na strežniku in ali JVM, kljub temu je mogoče napisati namizno Java aplikacijo. Za strežniške in ali JVM aplikacije, je za prehod na novejšo Java verzijo dovolj preprosti scenarij ponovnega prevajanja (recompile) in preverjanja hroščev v programski kodi.
- Preselitev obstoječe Visual Basic aplikacije v .NET.

Programski jezik Visual Basic .NET je v celoti prenovljen (objektno orientiran) in odpravlja veliko pomanjkljivosti predhodnika Visual Basic. Čeprav je Microsoft zatrjeval, da je mogoče z minimalnimi popravki preiti na novejšo verzijo Visual Basica, v praksi ponavadi ni tako. Ponavadi preprost scenarij ponovnega prevajanja in odpravljanja hroščev ni dovolj in je potrebno veliko ponovnega kodiranja. Prav tako so velike spremembe v grafičnem vmesniku IDE, ki se ga morajo na novo naučiti tudi izkušeni programerji.

Pri preselitvi aplikacije je potrebno narediti naslednje: preučiti redundantnost kode, identificirati objekte in ločiti podatkovni dostop aplikacije od uporabniškega vmesnika. Ko je enkrat narejena ta analiza, lahko opredelimo objektni model. Na podlagi tega modela lahko določimo potreben napor, ki ga bo potrebno vložiti v razvoj nove aplikacije. To predstavlja pravi čas za načrtovanje aplikacije, da bo takšna kot si jo želimo in ne na pol dobra programska rešitev.

Preselitev obstoječe Java aplikacije v .NET

.NET ne podpira Jave. Čeprav sta J# in C# programerjem Jave zelo podobna jezika, nista enaka Javi. .NET ne podpira JSP (Java server pages); zato je potrebno v predstavitvenem sloju konvertirati JSP v ASP.NET. .NET ima na razpolago veliko strežniških komponent, ki lahko zamenjajo večino funkcionalnosti shranjenih procedur in Java objektov, toda ne brez velike rekonstrukcije Java aplikacije. Če je videti to preveč zahtevno, si lahko pomagamo z dodatnimi orodji (third party products), s katerimi lahko premostimo čisto Javo in .NET.

Preselitev obstoječe Visual Basic aplikacije v Javo

Ta premik je najzahtevnejši. Vse obrazce, module, razrede, podatkovne tipe, itd. je potrebno rekonstruirati. Dostop do podatkovnih baz je potrebno konvertirati v JDBC in ActiveX v Java objekte. ASP strani je potrebno konvertirati v JSP (Java Server Pages) To pomeni, da je potrebno aplikacijo napisati popolnoma na novo. Kljub temu obstajajo orodja (ASP2JSP, VB Converter, Java Edition), s katerimi si lahko pomagamo pri konvertiranju.

Stroški razvijanja aplikacije

J2EE predstavlja idealno platformo za dobro izučene in izkušene programerje, ki znajo programirati na »nizkem« nivoju. Ti programerji so spretni v objektnem programiranju in imajo globoko znanje o razvoju Jave. Čeprav je Java edini programski jezik v J2EE, se lahko v J2EE uporabljajo aplikacije, ki so napisane tudi v drugem programskem jeziku, skozi CORBA.

.NET rešuje težavo programerjev v Visual Basicu, ki so se počutili nekoliko odrinjeni glede internetnega programiranja v primerjavi z Java programerji. Z .NET lahko programer razvija aplikacijo v svojem jeziku, .NET pa bo poskrbel za veliko podrobnosti na »nizkem« nivoju, vključujoč generiranje XML in interakcijo z SOAP-om. To omogoča hiter začetek za veliko razvijalcev programske opreme, ki so bolj spretni v svojem tradicionalnem jeziku. Kljub temu je potrebno upoštevati dejstvo, da več možnih programskih jezikov lahko povzroča določeno izpostavljenost aplikacij. Če dovolimo razvijanje aplikacije v več jezikih, nam lahko to pozneje povzroča nočne more glede vzdrževanja aplikacij.

Varnost

Osnovni varnostni model za obe platformi predstavlja omejitev dostopa do funkcij in delov kode. .NET definira omejitve dostopa do kode z CLR-jem v domeni aplikacije. V .NET izvajalnik kode uporablja dve vrsti varnosti: varnost kodnega dostopa in varnost, ki temelji na vlogah.

Izvajalnik kode varnostno preveri kodo, preden je izvedena in med JIT prevajanjem, tako da analizira metapodatke in primerja zahteve za dostop s sistemskimi nastavitvami. Ena od pomembnih komponent varnosti kodnega dostopa je preverjanje, ali je koda tipsko varna (type-save). Ko prevajalnik JIT pregleda metapodatke in MSIL ter ugotovi, da program ne skuša dostopati do pomnilnika zunaj svojega obsega, je koda tipsko varna. Varnost, ki temelji na vlogah, je zelo podobna varnosti, ki jo danes uporablja Windows. Ustvarimo lahko vloge (roles), ki jim dodelimo dovoljenja.

Marketinški prijemi

Microsoft je pognal celotni marketinški stroj za uveljavitev .NET tehnologije. Seveda Microsoft predstavlja svojo tehnologijo kot vodilno, ki je med uporabniki zelo razširjena in ima dominantni položaj na tržišču. Očitno je Microsoft najmočnejši v trženju in razglaševanju »svojih« novih idej.

Na drugi strani imamo industrijo, ki je enotna v podpori J2EE. Veliki industrijski prodajalci (BEA, IBM, Oracle in seveda SUN) so se odločili za podporo J2EE. Gre za veliki izziv in kdo bo zmagal v bitki za čim večji delež na trgu ostaja uganka.

Tabela 1: Osnovna primerjava J2EE in .NET

Lastnosti	J2EE	.NET
Stroški licence	+	+
Stroški razvoja aplikacij	-	+
Izgradnja spletnih storitev	+	+
Podpora prodajalca	+	+
Platformska neodvisnost	+	-
Platformska zrelost	+	-
Neodvisnost od programskega jezika	-	+
Podpora različnih proizvajalcev	+	-
Sposobnost integracije z obstoječimi aplikacijami	+	-
Neodvisnost naprav odjemalca	+	+
Orodja za hiter razvoj aplikacij	-	+
Napredne novosti	+	+
Najnovejše novosti	+	+
Enostavnost razvijanja aplikacij	-	+
Integracija orodij za razvijanje aplikacij	-	+
Integracija z obstoječimi aplikacijami	+	-
Stabilnost produkta	+	-
Prodajalčeva stabilnost	+	+
Zanesljivost	+	+
Varnost	+	+
Razpoložljivost	+	+
Podpora sistema	+	+
Dokumentacija produkta	+	+
Administrativna orodja	-	+
Enostavnost vzdrževanja	-	+

Vir: www.infoad.com

V tabeli 1 so označene pozitivne lastnosti s + in negativne z -. V primeru, da je označeno za prvo tehnologijo s + in za drugo z -, pomeni, da ima prva tehnologija prednost pred drugo. V primeru, da imata obe tehnologiji označeno lastnost s +, pomeni, da sta obe tehnologiji dobri glede te lastnosti. Obe tehnologiji imata določene slabe lastnosti, če je označeno z dvema minusoma.

Kot je razvidno v tabeli 1, ima tehnologija .NET prednost v manjših stroških razvoja aplikacij, ker ima boljše orodje za razvijanje programskih rešitev. Na nižje stroške razvoja aplikacij vplivajo tudi nižji stroški izobraževanja. Prednost .NET tehnologije je tudi v neodvisnosti od

programskega jezika, boljši integraciji razvojnega orodja z ostalimi programskimi orodji in v enostavnosti vzdrževanja.
J2EE tehnologija ima prednost v platformski neodvisnosti. Prav tako je ta tehnologija bolj preizkušena in nudi boljšo povezljivost z obstoječimi poslovnimi aplikacijami.

6.2. C# in Java

6.2.1. Zakaj je potreben nov programski jezik

Obstaja veliko debat o tem, ali je smiselno uvajati novi programski jezik, medtem ko imamo na razpolago že veliko obstoječih. Ocenjuje se, da je dovolj prostora še za en jezik. Navsezadnje predstavlja novi jezik samo novo orodje, ki ga lahko profesionalci uporabijo za izdelavo programskih rešitev. Na primer C++ je bil narejen kot izpeljanka C programskega jezika, in je v originalu bil imenovan »C with classes«. Čeprav je C inovativni in ekstremno učinkovit jezik, ima težave z zanesljivostjo, fragmentacijo kode in zelo kompleksnim upravljanjem spomina. Delno te težave odpravlja C++.

Prenosljivost kode med procesorji se razlikuje od prenosljivosti kode med različnimi operacijskimi sistemi. Različni operacijski sistemi dostopajo do podobnih sistemskih opravil različno. Vsak, ki je kdaj poskušal kreirati, na primer grafični vmesnik, ki je prenosljiv med platformami, razume problem prenosljivosti.

Java je bila napisana leta 1995. Odpravila je težave kompleksnosti, upravljanja s spominom in prenosljivosti med operacijskimi sistemi. Java je nastala tudi zaradi potreb podjetij, ki so želela izboljšati obstoječo strojno opremo, ki je bila do tedaj odvisna od določenega operacijskega sistema. Največji vpliv na razvoj Jave je prav gotovo imel razvoj interneta.

Glavni razlogi za uvedbo C#.
Glavni cilj, ki ga je želel doseči Microsoft s C#, je podpora za RAD (Rapid Application Development). Internetne aplikacije je potrebno razviti tudi z internetno hitrostjo, torej je zelo pomemben čas v katerem se lahko razvije aplikacija. Zato mora biti jezik enostaven za uporabo, ki se ga da hitro naučiti, producira kratko kodo in je enostaven za vzdrževanje. Medtem ko Delphi, Java in Visual Basic izpolnjujejo v neki meri te pogoje, C++ ni tako uspešen. C++ je kompleksni in okorni jezik za upravljanje in samo nekaj knjižnic omogoča preprosti vmesnik. Dodatno ima C++ težave z upravljanjem spomina, ki ga je potrebno manualno upravljati, kar povzroča marsikomu velike težave. C++ ne zna preprečiti težav z nezdružljivostjo različnih verzij. Kljub velikim naporom Microsofta in Borlanda, ni mogoče uporabiti C++ za hiter razvoj aplikacij.

Internetni programski jezik mora že po svoji definiciji podpirati platformsko neodvisnost. Ker internet predstavlja omrežje različnih naprav, mora servis zagotavljati delovanje široke palete

strojne in računalniške opreme. Še več, programi na strani odjemalcev, morajo biti zmožni, da tečejo na različnih napravah, vključujoč dlančnike in prenosne telefone. Takšna fleksibilnost predstavlja pravi izziv za vse programske jezike, z izjemo Jave. Visual Basic je v osnovi namenjen samo za Windows aplikacije; tudi Delphi trpi za določenimi omejitvami; zato je v razvoju Delphi za Linux. Microsoft s C# želi doseči boljšo podporo za COM in DCOM. COM potrebuje že dalj časa močno podporo programskega jezika. Večina jezikov, C++ in Delphi zahtevajo, da programerji kreirajo dodatno razrede in posebne dodatke za vsak COM objekt, ki ga kreirajo.

Nekateri prodajalci so izdelali določene čarovnike, ki bi naj olajšali COM in OLE opravila, vendar ti pripomočki ne zakrijejo v celoti kompleksnosti. Visual Basic zna uspešno prekriti kompleksnost COM, vendar na račun hitrosti izvajanja. Visual Basic nima samo pomanjkljivost v hitrosti, ampak tudi v tem, da ni v celoti objektno orientiran jezik. Vse te tri cilje je Microsoft uspešno dosegel s C#. Lahko ugotovimo, da C# združuje vse dobre lastnosti Jave in Delpija na eni strani in C ter C++ na drugi strani.

6.2.2. Koliko sta si Java in C# podobna

Sintaksa C# je skoraj identična Javi; vsebuje značilnosti sodobnih programskih jezikov, kot je na primer dedovanje razredov. Vendar se C# razlikuje od Jave v obliki jezika, ki je sposojena od programskega jezika Delphi in direktni integraciji z objektnim modelom COM (Component Object Model).

Slika 24: Enostavna primerjava kode Java in C#

Java	C#

public class GlobalGreeting {
  public static void main(String[] args) {
  System.out.println("Hello, world!");
  }
}
	class GlobalGreeting {
	  static void Main(string[] args) {
	  System.Console.WriteLine("Hello, world!");
	  }
	}

Vir: http://www.javaworld.com/jw-11-2000/jw-1122-csharp1-p2.html

Podobnost med tema dvema deloma kode je očitna. V obeh primerih je koda ovita v *main* funkcijo, ki je statična, ograjena z razredom. V obeh primerih je potrebno dostopati do globalnega imena *system*. Podobnost ni samo v izvorni kodi. Kot smo že omenil, Java pretvori z JVM vse ukaze v binarno obliko. C# podobno prevede vse ukaze v MSIL (Microsoft Intermediate Language).

Zunanja referenca

Koda za uporabo zunanjih modulov je prav tako podobna v Javi in C#. Java uporablja rezervirano besedo *import*, za deklaracijo reference, C# uporablja za isti namen *using*.

Slika 25: Primer zunanje reference

Java	C#

import java.lang.System;
public class GlobalGreeting2 {
  public static void main(String args[]) {
  System.out.println("Zdravo, zemjata!");
  }
}
	using System;
	class GlobalGreeting2 {
	  static void Main(string args[]) {
	  Console.WriteLine("Salut, monde!");
	  }
	}

Vir: http://www.javaworld.com/jw-11-2000/jw-1122-csharp1-p2.html

Obe rezervirani besedi delujeta na isti način; v obeh primerih lahko uporabimo samo ime modula in ni potrebno v celoti specificirati referenco. Razlika med »*import*« v Javi in »*using*« v C# je v tem, da Java ima paketni koncept, medtem ko C# uporablja »*namespace*«, ki vsebuje statično vse globalne metode (npr. System.Console.WriteLine()).

Oba jezika pa imata zelo podobno sintakso jezika C++. Javi manjka ukaz »*foreach*«, ki se uporablja za iteracijo kolekcije. Podobnost C++ in C# predstavlja veliko pridobitev za vsako podjetje, ki je že veliko vložilo v izobraževanje kadrov. Izkušeni programerji v C++ ne bodo imeli večjih težav z razumevanjem C#. C# zagotavlja Windows razvijalcem enostavnost programiranja, kot jo ponuja Java, obenem pa lahko programirajo direktno za Windows platformo.

Največja pridobitev za Windows programerje s C# je enostavna integracija z obstoječo Win32 komponentno tehnologijo – COM. Microsoft je omogočil, da lahko razvijamo komponente v različnih jezikih. Razvijalci programske opreme lahko izbirajo različne jezike za reševanje različnih problemov. Še bolj pomembno je, da se ni potrebno programerjem učiti novega jezika. Vendar imamo lahko ob vsem tem določene pomisleke, kot na primer:

- Ali si želimo upravljati projekt, v katerem programerji razvijajo aplikacijo v različnih jezikih. Tudi če smo strokovnjak, ki obvlada vse te jezike, se postavi vprašanje ali bi želel naš vodja voditi takšen projekt.
- Kaj narediti s programerji, ki na vsak način vztrajajo pri svojem programskem jeziku, pa čeprav so to eksotični jeziki; na primer: APL, Haskell, Prolog, itd. Vprašanje je, ali je smiselno osebje izobraževati s toliko različnimi jeziki.

- Zamislimo si, da smo razvili projekt s šestimi različnimi programskimi jeziki. Postavi se vprašanje, kako bomo vzdrževali kodo, če eno izmed podjetij, ki je razvilo jezik, propade.
- Kaj, če se odločimo, da bomo spremenili platformo? Lahko se zgodi, da bomo nekega dne ugotovili, da Microsoftovi produkti ne zadovoljijo naših potreb. Lahko se zgodi, da določeno tehnologijo vsekakor želimo uporabiti, kljub temu, da jo .NET ne podpira.

Navsezadnje se postavi vprašanje, kako bomo vzdrževali različne verzije programskih jezikov. Res je, da se programski jeziki spreminjajo počasneje kot programi, kljub temu pa se spreminjajo. Če npr. razvijemo del kode v JScript.NET in del v C#, potem pa razvijalci v jeziku C# odidejo, se mora ekipa, ki je pisala v JScriptu, naučiti jezika C# in dešifrirati obstoječo kodo. Kot prednost prilagodljivosti vsem jezikom pa vzemimo scenarij, kjer imamo npr. omejen proračun, v ekipi pa ljudje poznajo različne jezike. Vsi lahko takoj začnejo uporabljati .NET in se jim ni treba učiti skupnega jezika.

V realnosti ni vse tako črno - belo. Komponente napisane v različnih jezikih so vključene v mnoge velike sisteme. Večina strežnikov podpira vse glavne programske in skriptne jezike. S to raznolikostjo lahko dosežemo potrebno fleksibilnost in moč. Vendar samo ob pogoju, da so jeziki izbrani s tehtnim premislekom glede na izbrano arhitekturo.

6.3. Splošna primerjava

Prodajalci tehnologij in organizacije poskušajo prilagoditi svojo infrastrukturo, ki bi bila primerna za elektronsko poslovanje. Istočasno s spremembo poslovnih zahtev, sta se na tržišču pojavila dva glavna aplikacijska razvojna modela: Java 2 Enterprise Edition (J2EE) in .NET. J2EE predstavlja strežniški aplikacijski model, ki ga je v osnovi razvil Sun Microsystems. .NET je prav tako strežniški aplikacijski model za platformo Microsoft Windows.

Oba aplikacijska modela imata podobno arhitekturo, vendar se bistveno razlikujeta v filozofskem ozadju – J2EE cilja na široko uporabo odprtih standardov; .NET cilja na samostojni operacijski sistem Windows.

Katero tehnologijo izbrati ni lahka odločitev. Za kratkoročne rešitve sta obe aplikacijski arhitekturi primerni. Za dolgoročne rešitve pa je potrebna zelo premišljena odločitev. Napačna odločitev lahko predstavlja za podjetje velike stroške in izgubo konkurenčne prednosti. Na splošno velja ugotovitev, da aplikacije, ki bazirajo na internetnem brskalniku, so stroškovno ugodne. Za zmanjšanje TCO (Total Cost of Ownership) je potrebno izbrati samo eno tehnologijo. Če poskusimo uporabiti obe, bo mnogo težje doseči efektivnost, ki jo ponuja samostojna arhitektura.

Če želimo ti dve arhitekturi pravilno primerjati, je potrebno primerjati med seboj primerljivo. J2EE je specifikacija z velikim številom produktnih implementacij, .NET je pa mešanica

specifikacij in izdelka. Zato je potrebno med seboj posebej primerjati specifikacije in izdelek. Oba J2EE in .NET uporabljata (runtime) izvajalno okolje na strani strežnika in številne izvajalne storitve za aplikacije. Na višjem nivoju sta si obe tehnologiji precej podobni, kljub temu pa se v podrobnejši analizi bistveno razlikujeta.

Specifikacija

Najprej je potrebno definirati kaj bomo sploh primerjali. J2EE je na edinstven način specifikacija, ki je implementirana s strani večjega števila prodajalcev programskih rešitev. Mnogo prodajalcev je te rešitve vgradilo v svoje produkte. Na primer Oracle, Sun, IBM in BEA ponujajo zelo konkurenčne aplikacije, ki bazirajo na J2EE. J2EE predstavlja razširjen del Java platforme, ki je razdeljena v tri izdaje: Java 2 Standard Edition (J2SE), Java 2 Enterprise Edition (J2EE) in Java 2 Micro Edition (J2ME). Specifikacije so se v teh primerih razvijale skozi standardni proces, imenovan Java Community Process. V tej skupnosti je vključena večina prodajalcev, ki ne podpirajo Microsoftovih rešitev iz različnih razlogov.

.NET je vzdevek za veliko novih stvari v Microsoftu. Na programskem nivoju, to predstavlja strežniški model s podobnimi karakteristikami kot J2SE, J2EE in J2ME. V primerjavi z J2EE nas nekoliko zmede naziv Microsoft .NET, ki je tudi blagovna znamka za Microsoftove produkte, ki se ponujajo za izgradnjo na višjem nivoju (nad programskim okvirjem in Windows operacijskem sistemom).

Primerjava ogrodja

Bolj smiselna je primerjava v širšem pogledu, to je primerjava celotne platforme, in ne samo posameznega dela specifikacije. To lahko storimo na štirih nivojih: Aplikacijski model, implementacija strežnika, ponudba izdelkov in spletne storitve kot je prikazano v Tabela 2.

Tabela 2: Primerjava platform

Primerjalni nivo	Java	Microsoft
Aplikacijski model	J2EE specifikacija	.NET okvir (framework)
Implementacija strežnika	Različni prodajalci	Microsoft
Ponudba produkta	Več prodajalcev	Več prodajalcev
Spletne storitve	Več prodajalcev	Več prodajalcev

Vir: Stoeckert, 2003

Ko primerjamo J2EE in .NET na visokem nivoju, je pomembna ugotovitev, da J2EE strežniška implementacija bazira na večjem številu prodajalcev in s tem dokumentiranih specifikacij. .NET predstavlja rešitev enega prodajalca, Microsofta, ki ponuja .NET strežniško implementacijo bazirano na enem okvirju (angl. framework). Ne tako očitno, toda zelo

pomembno je, da je J2EE specifikacija razvita in upravljana na odprtem standardu (Java Community Process). Pri .NET-u razvija in upravlja jedro .NET specifikacije samo en prodajalec – Microsoft.

Smo torej pred odločitvijo, če izberemo .NET smo izbrali Microsoft. Če izberemo J2EE pomeni, da smo izbrali tržišče na katerem je več prodajalcev, ki so med seboj konkurenčni. Tekmujejo v zmanjševanju stroškov, zmogljivostih in ostalih karakteristikah. Vsega tega pri .NET ni.

6.4. Arhitektura

J2EE in .NET bazirata na konceptu n-slojne arhitekture. Smisel te arhitekture je v tem, da ločimo uporabniški vmesnik od poslovne logike, ki je ločena od podatkovnih zbirk in infrastrukture. Ta tip arhitekture temelji na srednjem sloju (middle tier) vsebnika, ki zagotavlja številne storitve za zagotavljanje zanesljivosti, razpoložljivosti in zanesljivosti. Odvisno od implementacije, vsebnik tipično ponuja zbir »bogatih«storitev za varnost, transakcije in povezovanje. Obe J2EE in .NET arhitekturi zagotavljata razvijalcem že to jedrno infrastrukturo in je razvijalcem ni potrebno graditi. Pomembno je, da razumemo način programiranja, da dosežemo skladno arhitekturo.

Mnoge obstoječe aplikacije, ki še tečejo na glavnem računalniku zahtevajo proceduralno programiranje. Podatkovne strukture so medsebojno odvisne in procedure so narejene za manipulacijo s podatki. S tem, ko narašča kompleksnost programov, se povečuje tudi odvisnost podatkovnih struktur in procedur. Zato z naraščanjem kompleksnosti postaja vse težje vzdrževanje in podpora teh aplikacij.

Objektno orientirano programiranje (Object Oriented Programming - OOP) omogoča učinkovitejše programiranje razvijalcem. Z objektno orientiranim programiranjem, objekti vsebujejo stanja in svoje okolje, ki dovoljuje samostojni entiteti, da opiše kaj je in kaj lahko počne. Objekti so relacijsko povezani med seboj. Tudi OOP uporablja koncept enkapsulacije in polimorfizma. OOP poenostavi kompleksno programiranje, kodiranje, vzdrževanje in podporo.

Današnje storitveno objektno programiranje (Service Object Programming - SOP) predstavlja še bolj sofisticirano metodo razvijanja aplikacij, s katero lahko učinkovito ponovno uporabimo programsko kodo, zagotovimo manj napak in povečamo zmožnost uporabe v prihodnosti. V SOP-u je aplikacija opisana mnogo bolj naravno. Vsaka funkcija aplikacije je lahko kandidat za storitev.

Storitev predstavlja del jedra poslovne logike in je protokolno neodvisna. Ne vsebujejo predstavitvene logike in tudi ne logike za integracijo s podatkovnim slojem, kot so podatkovne zbirke. Storitve so osredotočene izključno na reševanje problemov v poslovni domeni. Storitve

so zelo ohlapno povezane z ostalimi komponentami v aplikaciji. Ker so protokolno neodvisne, je mogoče do iste storitve dostopati na različne načine. To dovoljuje storitvam, da izvedejo svojo poslovno logiko in vrnejo rezultat v enem samem klicu. Na primer, storitev imenovana getZnesek lahko vrne informacijo za vse bančne račune določenega uporabnika.

6.4.1. Primerjava arhitektur

Na višjem nivoju sta obe arhitekturi podobni. Obe temeljita na raziskavah iz osemdesetih in devetdesetih let, na več slojni arhitekturi. Kljub temu so očitne razlike, ko pogledamo arhitekturi podrobneje. V nadaljevanju se bomo osredotočili na razlike v: prenosljivosti, arhitekturnih specifikacijah, programskih jezikih, globini arhitekture in spletnih storitvah.

Prenosljivost

Pri analizi arhitekture lahko odkrijemo očitno razliko med J2EE in .NET. J2EE podpira različne platforme (tudi Windows), .NET podpira samo eno, Microsoft Windows. J2EE podpira široko področje platform vključujoč: Sun Solaris, HP Unix, IBM AIX, IBM OS 390 in celo Linux.

Zaradi potrebe po racionalizaciji stroškov (ROI - Return On Investment) v odjemalec/strežnik arhitekturi poskušajo prodajalci najti nove možnosti uporabe terminala. Tako so razvili »pametni« terminal t.i. Sun rays. Ta terminal je poznan tudi pod imenom »grafično neumni terminal« ali »ultra« lahki uporabniški strežnik ki ne vsebuje procesorja in trdega diska. Odprta koda je glavna alternativa Microsoftovim namiznim aplikacijam. Sum Microsystems in Oracle sta najbolj zaslužna za napredek v skupnosti odprte kode. Tehnologije, kot so Mozzila, Star Office, JXFS (Java Extensions for Financial Services) predstavljajo še zadnje koške v zloženki za poslovne aplikacije in finančne storitve. Skupni strošek lastništva (TCO – Total Cost of Ownership) lahko s temi terminali zmanjšamo tudi do 75% v primerjavi z obstoječimi Windows aplikacijami. Z uporabo terminalov Sun Rays zmanjšamo tudi porabo energije, odpravimo grožnjo virusov in zmanjšamo stroške administracije.

Pomembno je, da uporabimo kreativne rešitve obenem pa ne pozabimo na že obstoječe aplikacije. J2EE tehnologija je pri tem zelo uspešna, ne podpira samo novih tehnologij, temveč tudi obstoječe. V nasprotju .NET podpira samo eno platformo, Microsoft Windows. Kljub začetnim obljubam, da bo .NET podpiral vse pomembnejše platforme, se je izkazala ta trditev za napačno. V letu 2003 se je Microsoft odločil, da bo podpiral samo Windows platformo.

Specifikacija arhitekture

Razvijalci programskih rešitev lahko poiščejo specifikacije posameznih komponent na spletni strani Java skupnosti (http://www.jcp.org). V JCP (Java Community Process) je aktivnih več kot 500 članov (individualni in podjetja), ki delajo na izboljšavah J2EE specifikacij in novih vmesnikov. Te specifikacije nato uporabljajo razvijalci, da razumejo arhitekturo in prodajalci,

da implementirajo to arhitekturo v izdelke. Pomembno pri specifikacijah je tudi to, da ima vsaka specifikacija referenco implementacije. S tako porazdeljeno skupino razvijalcev je mogoče zbrati več znanja, kot pri .NET, ki predstavlja zaprti proces.

Naslednji pomislek, ki ga je potrebno upoštevati, predstavlja zrelost platforme. Spletne storitve morajo biti zanesljive, združljive in zelo zmogljive. Kvaliteta spletnih storitev temelji na vseh slojih platforme, od strojne opreme, operacijskega sistema, do spletnih komponent in upravljavskih orodij. Ustanovitelj Jave, Sun Microsystems, je že od nekdaj videl svet kot model storitve in je zgradil zrelo, preizkušeno in stabilno platformo v J2EE. Microsoft je v tem pogledu novinec.

Potrebno je tudi upoštevati, da je .NET produkt enega prodajalca. J2EE predstavlja tržišče, na katerem deluje s partnerji, razvijalci in prodajalci. Na primer, mnogo svetovalnih podjetij predstavlja hrbtenico J2EE, tako, da stranke ne dobijo samo robustno infrastrukturo spletne storitve, ampak tudi široki spekter svetovalne podpore. Pomislimo tudi na zagon, ki poteka v ozadju procesov odprte skupnosti. Ko nekaj programerjev z lahkoto zgradi in prilagodi košček programske opreme, se zgodijo osupljive inovacije in izboljšave. Enako velja tudi za J2EE, ker zahteve prihodnosti niso diktirane od samo enega prodajalca, kot je v primeru .NET, ampak jih diktira celotni trg skozi JCP.

V nasprotju, je .NET arhitektura dokumentirana znotraj ponudbe Microsoftovih produktov. Referenčno implementacijo predstavlja sam Microsoft .NET. V nasprotju z lastniško percepcijo, se je Microsoft odločil, da del .NET okvirja objavi v ECMA (European Community Standards Association). Microsoft ni želel objaviti celotno standardizacijo v Java skupnosti. Ocenjuje se, da je Microsoft objavil le 10% .NET okvirja.

Programski jezik

Naslednja opazna razlika med tema dvema arhitekturama je v programskem jeziku. Microsoft trenutno podpira .NET s sedemintridesetimi jeziki. V J2EE okolju se uporablja samo programski jezik Java in ostali jeziki s pomočjo določenih API vmesnikov, kot je npr. Java Connector Architecture. V .NET sta dejansko glavna dva jezika: Visual Basic .NET in C#. Visual Basic .NET je Microsoftov najpopularnejši jezik, katerega je v celoti prenovil in je sedaj v pravem smislu tudi objektni jezik. C# je posebno prilagojen za nižji nivo programiranja, vendar ne tako kompleksen kot je C++.

Java je zelo razširjena. Ocenjuje se, da jo uporablja več kot dvajset milijonov programerjev (od tega 2,5 milijona profesionalnih razvijalcev programske opreme). Popularnost Jave ni narasla samo z razvojem J2EE arhitekture, ampak tudi v izobraževalnih sistemih po celem svetu.

Kot smo že v predhodnem poglavju omenili, se v .NET uporablja CLR za podporo različnim jezikom. Glede CLR-ja poteka veliko debat. Dejansko predstavlja CLR mojstrsko delo, vendar po drugi strani še ni dovolj preizkušen v zelo zahtevnih sistemih. Obstajata dve področji za zaskrbljenost z CLR-jem: upravljana koda v primerjavi z neupravljano kodo in vsesplošna zmogljivost in razpoložljivost.

Da bi Microsoft zagotovil uporabo tako velikega števila programskih jezikov, je moral odstraniti mnogo splošnih karakteristik in podatkovnih tipov iz posameznih jezikov. Microsoft ponuja kot rešitev uporabo koncepta neupravljane kode. Neupravljana koda omogoča dostop do izvirnega jezika in istočasno združljivost z .NET-om. Vse to lahko predstavlja veliko tveganje, če se bomo v prihodnosti odločili za posodobitev .NET okvirja.

Mogoče je še pomembnejša zaskrbljenost glede zmogljivosti in razpoložljivosti. Jasno je, da je CLR popolnoma nova tehnologija in je zato malo pravih življenjskih izkušenj v zelo zahtevnih okoljih. Potrebno bo več let, preden bo dovolj ekspertov na razpolago za izdelavo zahtevnih rešitev. Začetni testi so v neki meri razočarali v pomanjkanju zanesljivosti .NET spletnih storitev. Nekateri prodajalci so te težave rešili z dodatnimi rešitvami na višjem nivoju (implementacija med sistemi). To pa seveda predstavlja dodatno kompleksnost rešitve in s tem povezane večje stroške (TCO).

V nasprotju je JRE (Java Runtime Engine) na tržišču že več kot osem let. V tem času je ta tehnologija že zelo dobro preizkušena, tudi na različnih platformah, in kar je zelo pomembno, nenehno se izboljšuje z novimi izkušnjami ekspertov. Zahteve po zmogljivosti niso določene samo z najboljšo prakso upravljanja JRE-a, ampak tudi zaradi konkurence med JRE ponudniki na tržišču.

Globina arhitekture

Na nivoju podatkovnega povezovanja, J2EE specifikacije ponujajo številne programske vmesnike API-je vključujoč: JDC (Java Database Connectivity) za podatkovne baze, JMS (Java Messaging Service) za asinhrono povezovanje in JCA (Java Connector Arhitecture) za poslovne informacijske sisteme. Microsoft ponuja mešanico tehnologij kot je MSMQ (Microsoft Message Queuing) in produktov kot je Microsoft Biztalk.

Za razliko od .NET okolja, kjer imamo možnosti za implementacijo samo od enega prodajalca, je J2EE platforma narejena v tekmovalnem okolju prodajalcev, ki ponujajo različne implementacije in dodane vrednosti za produkte na teh specifikacijah. To tržišče poganjajo t.i. paketne aplikacije prodajalcev, kot so: Oracle, SAP in Peoplesoft. Vsi ti prodajalci so posvojili Javo v svojih arhitekturah. Vse to nam prikazuje določeno kapaciteto J2EE arhitekture, ki se nagiba k bolj razširjenemu opisu, dokumentiranju in implementaciji kot Microsoftova platforma. Do določene stopnje nam ta širša povezanost in interoperatibilnost kapacitet

predstavlja bolj porazdeljen niz prodajalcev v Java skupnosti, kot udeleženci, ki so vključeni v razvoj Microsoftovih produktov.

Integracija in spletne storitve

Mnogo organizacij je zelo zainteresiranih za razvoj spletnih storitev. Zato so pripravljene vlagati v ta razvoj znatna sredstva. S ponudbo XML baziranega dostopa v zaledne poslovne aplikacije, imajo organizacije nov in stroškovno zelo učinkovit način uporabe njihovega investiranja v tehnologijo. Na začetku so bile spletne storitve deležne mnogih kritik glede varnosti. Mnogi sistemi, ki so integrirani v druge sisteme, ne uporabljajo spletnih storitev prek HTTP-ja. Integracija sistem-sistem mora biti implementirana z uporabo: HTTPS, VPN, digitalnega certifikata ali kombinacijo teh treh. Vse skupaj je odvisno od tega, koliko slojev želimo uporabiti v skupni varnosti.

S tako množico spletnih storitev na horizontu potrebujemo inteligentno odločitev o tem kakšno storitev sploh ponuditi. Oba J2EE in .NET sta ponudila strežniško tehnologijo za izgradnjo poslovnih aplikacij. Oboji se strinjajo, da so SOAP, WSDL in UDDI pravilne rešitve za uspešno spletno storitev.

Uspešna vzpostavitev spletne storitve je odvisna od mnogo dejavnikov – obstoječih sistemov, relacij s strankami in partnerji, izkušenj od razvijalcev itd. Zaskrbljeni smo lahko, da bo Microsoft poskusil razširiti infrastrukturo .NET spletnih storitev na lastniški način in s tem zapreti uporabnike v okolje Microsoft. Veliko je zaskrbljenosti glede projekta Longhorn[17]. Z upoštevanje lastniških nagibanj na tem področju, se je ustanovila organizacija WS-I (Web Services Interopatibility Organization), ki je nevtralno telo za certificiranje implementacij spletnih storitev. Prodajalci kot so Oracle, IBM in Microsoft so ustanovili WS-I, da bi zagotovili specifikacije ali standarde na področju spletnih storitev. V organizacijo WS-I je vključeno približno 130 članov, od tega sedemdeset odstotkov prodajalcev in trideset odstotkov končnih uporabnikov.

6.5. Kriteriji za odločitev

Do sedaj smo pri primerjavi bili osredotočeni na osnovne tehnične značilnosti obeh arhitektur J2EE in .NET. Poleg tehničnih kriterijev je potrebno upoštevati tudi poslovne kriterije.

[17] Longhorn bi naj leta 2006 nadomestil XP operacijski sistem. Na voljo so že preizkusne različice. Presenetljiva je odločitev Microsofta, da umakne nekatere revolucionarne spremembe iz napovedanega sistema. Sistem bi naj bil stabilnejši, varnejši in zanesljivejši, saj temelji koda na Windows Server 2003 Service Pack 1. Zanimivo je tudi dejstvo, da bosta na trg prišla dva podsistema, Indigo in Avalon. Več informacij lahko najdete na naslovu: http://msdn.microsoft.com/Longhorn.

6.5.1. Poslovni kriteriji

Organizacija

Vsaka organizacija ima že obstoječe okolje, ki se mora integrirati v nove arhitekture. Pretekle izkušnje, obstoječa vlaganja, predlagana nova tehnologija in ostali faktorji igrajo pomembno vlogo pri končni odločitvi. Tehnologiji J2EE in .NET imata svoje specifične karakteristike in glede na okolje v organizaciji, je ena tehnologija lahko primernejša od druge.

V homogenem tehničnem okolju so organizacije bolj naklonjene k določeni posamezni tehnologiji. Če v organizaciji temelji celotna infrastruktura na Windowsih, je nadgradnja v .NET povsem logična izbira. V nasprotju, če je organizacija homogena v Unix platformi in heterogena prek različnih operacijskih sistemov in strojnih platform (zmožnost Jave, da teče na vseh platformah), je prav gotovo J2EE logična izbira.

Zanimivo je, da tudi razvijalci v Windows okolju ne gledajo na J2EE s prezirom. Vsaki J2EE prodajalec ponuja implementacijo strežniške aplikacije na Windows in Linux platformi, ker je Intelova strojna platforma stroškovno ugodna izbira. J2EE prenosljivost se je premaknila iz strežnika na uporabniško platformo. To lahko pripišemo zrelosti odprte kode.

Podpora prodajalcev

Možnost, da izberemo in zamenjamo prodajalca je pomembna karakteristika poslovne vrednosti arhitekture. Brez konkurenčne ponudbe ima organizacija zvezane roke do določenih dobaviteljev tehnologije; brez druge izbire kot ponovno zgraditi celotno arhitekturo. Microsoft je obljubljal, da bo .NET možno uporabljati tudi na Unix in drugih platformah in ne samo na Windowsih. Interesi znotraj Microsofta so preprečili razvoj .NET-a tudi za ostale platforme.

Aspekt prenosljivosti je atraktivni izraz pri izbiri prodajalca. Predstavlja pomemben argument pri pogajanjih, s katerimi poskušamo doseči najboljšo funkcionalnost za določeno ceno. Izbira prodajalca igra tudi pomembno vlogo v smislu investiranja v celotno okolje razvoja aplikacije. Pri izbiri .NET-a imamo na izbiro samo enega prodajalca in en operacijski sistem za razvoj aplikacije. Rezultat tega je samo ena možna cena – Microsoftova.

Pri izbiri J2EE se uporabi ponavadi naslednji scenarij: najprej razvijamo in testiramo aplikacijo na cenejšem okolju (Intel bazirana strojna oprema), nato preizkusimo aplikacijo na različni strojni opremi in operacijskih sistemih. Organizacija lahko izbira med različnimi cenovnimi razredi, ugotovi koristi obstoječih vlaganj in prikroji vlaganja v infrastrukturo glede na dejanske potrebe.

Razkošje platforme

V večini primerov se izbere enaki prodajalec za infrastrukturo (J2EE ali .NET) in tudi produkte, ki so za to infrastrukturo. Pomembni sta dve dejstvi pri primerjavi J2EE in .NET:

- Primerjava med arhitekturama J2EE in .NET se razširi tudi na nivo produktov. Prodajalci kot so Oracle, Sun in BEA ne tekmujejo med seboj le na nivoju infrastrukture, ampak tudi z novostmi produktov, ki jih ponuja Microsoft.
- Prodajalci platform v J2EE okolju tekmujejo med seboj tudi na širšem tržišču, z ostalimi J2EE kompatibilnimi produkti. Ta nivo tekmovalnosti na J2EE tržišču zahteva nenehne inovacije in izboljšave. Pri arhitekturi, ki jo nadzira samo en prodajalec ta tekmovalnost ni tako izrazita. Pozitivno pri vsem tej je, da so prodajalci prisiljeni ponujati nenehne izboljšave, če želijo obstati na tržišču.

Razpoložljivost resursov

Sama arhitektura je premalo, če nimamo izkušenega kadra za razvijanje, testiranje, in upravljanje te arhitekture. Oba J2EE in .NET sta prinesla znatne resurse na to področje. J2EE arhitektura se uporablja že od leta 1997. Glede na zadnje Gartnerjeve raziskave, uporablja Javo v svojih aplikacijah več kot osemdeset odstotkov organizacij. Zaradi tako velike razširjenosti Jave lahko organizacije lažje najdejo izkušene resurse za J2EE arhitekturo.

Microsoft ima prav tako zelo veliko razvojno skupnost. Kljub temu je potrebno upoštevati dejstvo, da je z izdajo .NET razvojnega okolja prišlo do znatnih sprememb glede na predhodno arhitekturo. Zaradi tega je trenutno zelo težko najti izkušene razvijalce .NET arhitekture. Obstajajo svetovalna podjetja, ki so specializirana za J2EE in .NET arhitekturo. Ponavadi imajo ta podjetja podpisane pogodbe, ki jih obvezujejo na izbiro samo ene arhitekture. Široka posvojitev J2EE na različnih platformah, vključujoč Windows, lahko vpliva na to, da se bodo svetovalna podjetja raje odločila za J2EE zaradi večjega tržišča in maksimalne prilagodljivosti.

6.5.2. Tehnični kriteriji

Arhitektura

Odločitev za določeno arhitekturo mora temeljiti na natančni analizi kvalitete in storitev, ki so na razpolago v določeni arhitekturi. Obe arhitekturi sta odlični na področju poslovnih aplikacij, J2EE ima za sabo več let izkušenj in uspešnih implementacij, na drugi strani je .NET arhitektura, ki je privlačna z določenimi novostmi, tveganje pa predstavlja odvisnost od samo enega prodajalca – Microsofta. S tem ko je Microsoft izboljšal razpoložljivost, zanesljivost in zmogljivost svoji arhitekturi, so veliko pridobili Microsoftovi kupci. Kljub temu so Microsoftovi kupci odvisni od Microsofta za te kvalitativne izboljšave..NET platforma je naredila premik od PC baziranih Windowsov k internetno baziranim spletnim storitvam in brezžičnemu svetu. Vendar pa .NET ne more samostojno servisirati strategije razvoja interneta v prihodnosti. Nove izdaje .NET-a bodo v prihodnje konsolidirale arhitekturo. Največja prednost .NET-a je uporabniško prijazna arhitektura, po kateri na splošno slovi Microsoft. Na

mestih, kjer je mogoče opaziti pomanjkanje zanesljivost v .NET-u, odtehta to pomanjkljivost enostavnost implementacije.

Zrelost arhitekture

Zrelost arhitekture ima lahko veliki vpliv na uspešnost projekta. Pod zrelost arhitekture štejemo najboljšo prakso, visoko izkušene kadre in razumevanje splošnih problemov pri razvoju aplikacije. V tem pogledu ima znatno prednost J2EE, ki z več kot sedmimi leti razvijanja aplikacij in več kot 42 različnimi J2EE licencami predstavlja zelo zrelo platformo. Za povečanje zrelosti so pričeli J2EE strokovnjaki objavljati serijo »najboljših praks« (best practices), ponavadi imenovano J2EE Design Patterns, s katerimi se dokumentira splošno načrtovanje rešitev iz J2EE perspektive.

Microsoft .NET platforma je bila razvita v letu 2002. Od leta 2002 do 2004 ni bilo predstavljenih pomembnih novosti, bolj so se popravljale določene napake in varnostne zahteve. V začetku leta 2002 je bilo razvito edino razvojno okolje Visual Studio .NET. Kljub določenim impresivnim tehničnim dosežkom, je potrebno .NET označiti kot nezrelo platformo.

Inženirstvo

Podobnost med .NET in J2EE je v integriranem razvojnem okolju (IDE). IDE je osredotočen na hitro razvijanje aplikacij (RAD – Rapid Application Development), ki je primerno za razvoj oddelčnih tipov aplikacij. Microsoft je imel vedno zelo uspešna razvojna orodja. Z razvojem .NET-a je Microsoft že prej uspešna razvojna okolja ponovno prenovil. Visual Studio je tako načrtovan za vse spremembe v arhitekturi Microsoftove platforme.

V nasprotju z .NET-om, kjer dominira Visual Studio .NET, ima J2EE tržišče mnogo različnih razvojnih orodij. Večji prodajalci, kot so Oracle, Sun in Bea so izdali orodja, ki ne omogočajo samo integracijo J2EE, ampak tudi razširjene razvojne platforme, v katere so vključena orodja: odprte kode, 3rd party in J2EE. Ta orodja predstavljajo mešanico RAD-a in MDA-ja[18]. Orodja MDA so v J2EE prostoru bolj razvita kot v .NET-u. Obstajajo tudi orodja, ki združujejo .NET in J2EE.

Rezultat te intenzivne konkurence je precejšnja izenačenost na področju orodij za razvijanje aplikacij. Kljub temu lahko ocenimo, da ponuja J2EE platforma večjo izbiro in več cenovnih razredov v primerjavi z .NET platformo.

Sožitje platform

Mnoge organizacije bodo morale podpirati istočasno obe okolji J2EE in .NET. Ker sinergija med integriranimi platformami ne predstavlja prednosti v tem scenariju, so se nekateri

[18] MDA (Model Driven Arhitecture) predstavlja IDE vmesnik, ki bazira na objektnih modelih (UML).

prodajalci odločili za velika investiranja, da bi omogočila sožitje platform. Tako je npr. Oracle 10g načrtovan tudi tako, da je integriran v .NET okolje. To zmanjšuje stroške in povečuje razpoložljivost, zanesljivost in združljivost .NET okolja. Drugi prodajalci so naredili rešitve za aplikacijske strežnike, ki ne podpirajo delovanje z .NET okoljem. Ko pogledamo na TCO, nima smisla izbrati strežniško aplikacijo J2EE, ki ne podpira delovanje z .NET-om, ker je TCO večji, če izberemo rešitve različnih prodajalcev.

Integracija in spletne storitve

Obstoječe aplikacije predstavljajo najtežji problem pri izgradnji spletne storitve. J2EE ponuja cenovno ugodne načine za pomoč pri odpravljanju glavobolov, ko se odločamo za integracijo obstoječih aplikacij. S temi orodji je mogoče zmanjšati, če ne celo odpraviti, potrebo po popravljanju kode in vmesnikov. Rešitev tega problema je JCA (J2EE Connector Arhitecture). JCA se lahko pogovarja z obstoječimi poslovnimi informacijskimi sistemi. Če JCA vmesnik ne obstaja, ga je mogoče razviti. Mnogi prodajalci v J2EE okolju so razvili te vmesnike za uravnavanje integracije obstoječih sistemov. Microsoft z .NET-om ne zagotavlja določenih integracij z obstoječimi sistemi, zato tudi ne ponuja orodja, ki bi bilo podobno JCA-ju.

Glede na to, da J2EE in .NET timi si domišljajo, da za spletne storitve ni potrebna izbira ene arhitekture pred drugo, bi bilo razumno pričakovati, glede na interoperatibilnost spletnih storitev, da J2EE lažje komunicira z .NET-om. Zato mnogo prodajalcev poskuša z raznimi interoperatibilnimi testi spletnih storitev prepričati kupce, da so njihove spletne storitve hitrejše od Microsoftovih. Kljub temu, poslovne aplikacije, ki so ali niso izpostavljene spletnim storitvam, še vedno potrebujejo kvaliteto storitev, transakcije, varnost in ostale poslovne storitve, ki so odvisne od podložene arhitekture. Na strateškem nivoju so spletne storitve preprosto dodatni sloj na vrhu J2EE ali .NET odločitve, zato moramo obtežiti odločitev še z dodatni odločitvenimi kriteriji. Po ponderiranju je potrebno ponovno razmisliti katera arhitektura je primernejša. Bolje je ponovno razmisliti na tem mestu, kot pa nadaljevati na obstoječi izbiri, ki morda potrebuje številne dodatne kompleksne rešitve, da bomo na koncu imeli zanesljivo, razpoložljivo in varno spletno storitev.

6.5.3. Trendi v razvoju informacijskih tehnologij

Ko ugotavljamo primernost določene tehnologije za informacijske potrebe je potrebno upoštevati tudi trende v spreminjanju informacijskih sistemov. Smernice na informacijskem področju so kljub hitrim spremembam dokaj stabilne. Osnovne smernice so v elektronskem poslovanju, povezovanju, vnovičnem inženirstvu, sestopanju, organizacijskem izločanju in decentralizaciji.

Informacijska tehnologija bo tudi v naslednjih letih doživljala najhitrejšo rast in dinamične spremembe. V to smo lahko prepričani zaradi trendov globalizacije, ki so se pričeli v osemdesetih letih in se bodo nadaljevali tudi v prihodnjih letih, še posebno za določene

industrijske panoge. Globalizacija je predvsem posledica razvoja računalniške in telekomunikacijske tehnologije. Razvoj te tehnologije omogoča oblikovanje svetovnih elektronskih trgov, kjer kupci in dobavitelji prihajajo v neposredni stik. Globalizacija prinaša poleg določenih problemov tudi koristi. Globalizacija bo ustvarjala nove trge na katerih bo zelo velika konkurenca. Organizacije bodo morale graditi konkurenčnost na pridobivanju, procesiranju in obvladovanju informacij ter hitrejšem in boljšem prilagajanju na novo nastale situacije.

Razvoj interneta je omogočil, da so postali sistemi za elektronsko poslovanje dostopni tudi srednjim in manjšim podjetjem. V prihodnosti lahko pričakujemo, da se bodo telekomunikacije vsaj tako hitro razvijale kot v zadnjih letih. Večina uporabnikov bo imela širokopasovni dostop do interneta, kar bo še dodatno povečalo možnosti elektronskega poslovanja. Podjetja bodo spoznala, da je elektronsko poslovanje zanje edina alternativa. V porajajočem se digitalnem gospodarstvu se bodo podjetja prej ali slej soočila s konkurenco lokalnih, regionalnih in globalnih tekmecev.

Konkurenčno prednost lahko poslovni sistemi dosežejo na različne načine. Poleg naravnih prednosti, ustreznih oblik trženja, učinkovitih distribucijskih kanalov je pomemben dejavnik tudi ustrezni informacijski sistem. Ugotovimo lahko, da ni dovolj samo obdelava in analiza podatkov, ampak tudi povezanost informacijskega sistema z okolico. Povezovati bo potrebno različne informacijske sisteme med seboj.

V zadnjem desetletju so postale zelo popularne paketne programske rešitve kot so: SAP, Oracle ERP, PeopleSoft, JDEdwards, Siebel in Clarify. Omenjene ERP rešitve delujejo dobro samostojno, vendar ustvarjajo t.i. informacijske otoke. Tako se dogaja, da zaposleni prepisujejo podatke iz ene aplikacija (npr. SAP) v drugo aplikacijo, ker te aplikacije niso med seboj povezane. Rešitev tega problema je v integraciji poslovnih aplikacij EAI (Enterprice Application Integration).

Spletne storitve

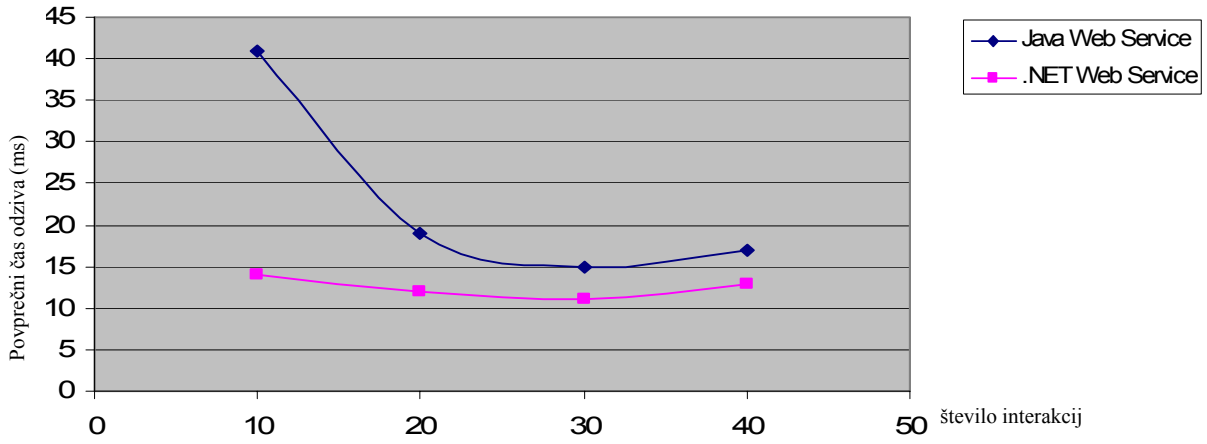
Pričakujemo lahko, da bodo v naslednjih letih zelo popularne spletne storitve. Raziskava Gartner[19] ugotavlja, da bodo podjetja do leta 2007 porabila milijardo dolarjev za dodatne licence programske opreme, ki jih potrebujejo za spletne storitve. Larry Peerlstein ugotavlja, da je sedaj pravi čas za pričetek poslovanja s spletnimi storitvami in da bodo tradicionalne oblike integracije še v prihodnje pomembne. Vzrok za takšno navdušenje nad spletnimi storitvami je prav gotovo v odprtem standardu, ki omogoča povezovanje različnih aplikacij ne glede na platformo. Za razliko od HTTP-ja, ki je omejen v opisu tipov podatkov, imamo na razpolago

[19] Gartner Dataquest, Maj 2003 (http://www.dataquest.com/press_gartner/quickstats/databases.html).

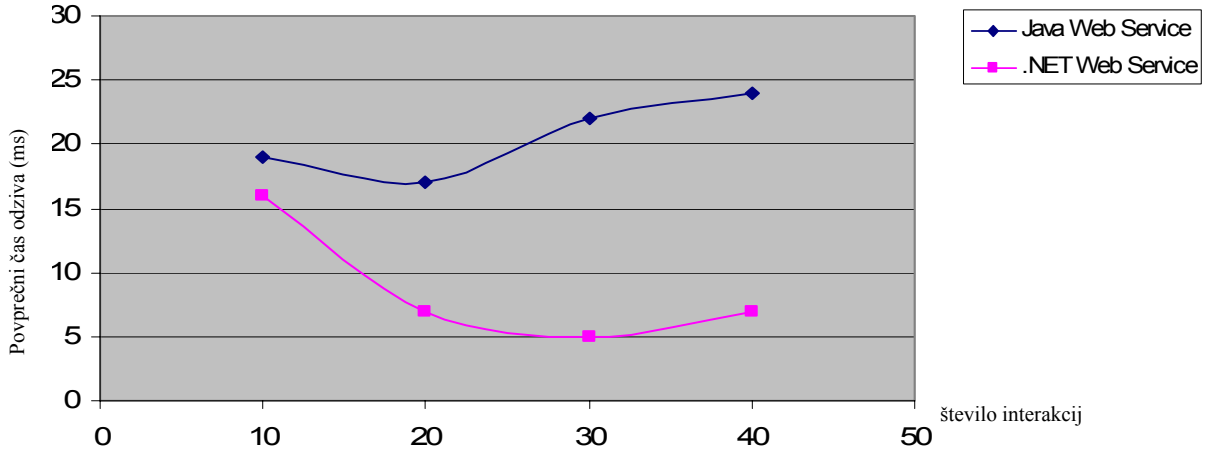
SOAP, ki lahko opiše vse vrste podatkov. V porazdeljenem izvajanju računalniških opravil je eden od glavnih izzivov oddaljeno klicanje procedur, ki je zelo zapleteno.

SOAP reši te težave na odprt in preprosti način, zato je logična rešitev za uporabo v spletnih storitvah. Spletne storitve bodo imele velik vpliv na porabnike in izdelovalce programske opreme. Pri uporabi spletnih storitev obstajajo še določene težave z varnostjo in preverjanjem pristnosti, ki pa bodo zagotovo v prihodnosti rešene. Razvoj spletnih storitev je z .NET-om enostavnejši in hitrejši kot z J2EE. Tudi po povprečnem času odziva spletne storitve, je .NET boljši od J2EE, kot prikazujeta sliki 25 in 26.

Slika 26: Primerjava zmogljivosti spletne storitve (20 istočasnih uporabnikov)
Test opravljen na strojni opremi: Centrino 1,8 GHz, 512 MB RAM-a.



Slika 27: Primerjava zmogljivosti spletne storitve (40 istočasnih uporabnikov)
Test opravljen na strojni opremi: Centrino 1,8 GHz, 512 MB RAM-a.



Vir: Rajgopalanm, 2003 (http://www.cis.ksu.edu/~vishwak/259).

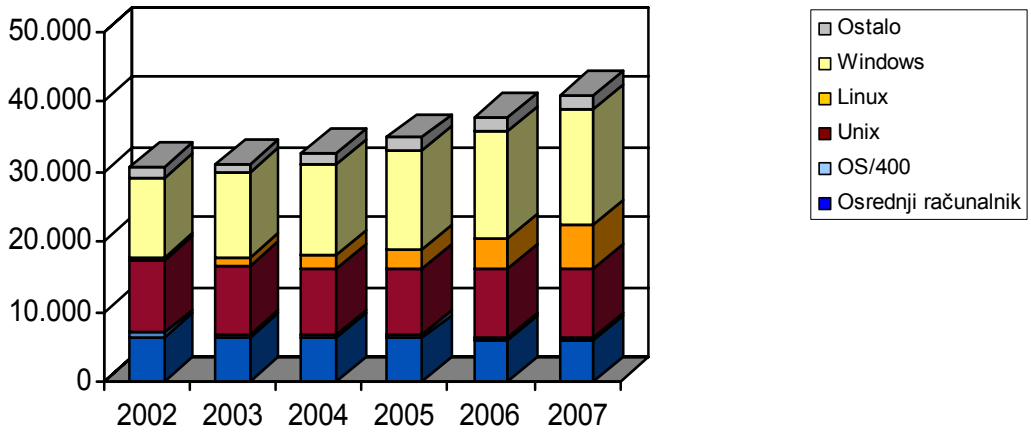
Večnivojska arhitektura

Večnivojska arhitektura odpravlja slabosti dvonivojske arhitekture. Primerna je za potrebe elektronskega poslovanja, kjer je potrebno zagotoviti zanesljivost delovanja pri visoki stopnji obremenitve. Trend razvoja programskih rešitev je v razvoju večslojne arhitekture. Ta arhitektura nam lahko zagotovi univerzalni dostop do poslovnih virov ne glede na uporabljen komunikacijski kanal ali napravo. Izboljšamo lahko zanesljivost, neprekinjenost delovanja in varnost. Prilagodljivost sistema je lahko z večnivojsko arhitekturo bistveno boljša. Posebnega pomena je odprtost tehnološke arhitekture, ki zagotavlja visoko raven povezljivosti s sistemi, ki temeljijo na različnih tehnologijah. To je ključnega pomena za doseganje konkurenčnih prednosti. S stališča uvajanja novih tehnologij je pomembna tudi razširjenost tehnološke arhitekture, ki omogoča hitro uvajanje novih sistemov ob sprejemljivih stroških. Iz tega lahko sklepamo, da bodo podjetja v prihodnosti iskale informacijske rešitve, ki bodo stabilne in povezljive in neodvisne od platforme. Neodvisnost od platforme je prednost J2EE tehnologije.
Potreba po hitrem razvoju programskih rešitev se bo še naprej povečevala. Večslojna arhitektura omogoča ponovno uporabo programskih rešitev. Funkcionalnost ločenih nivojev omogoča enostavnejše uvajanje novih programskih rešitev.

Trend prodaje

Odločitev o izbiri tehnologije J2EE ali .NET bo v veliki meri odvisna od platforme, ki jo podjetje uporablja ali jo bo uporabljalo v prihodnosti. Podjetja, ki uporabljajo osrednji računalnik, Unix in Linux operacijski sistem, se bodo verjetno odločila za J2EE. Na sliki 27 je razvidno, da se bo v prihodnje povečal delež Windows platforme, delež Unix platforme pa se bo zmanjšal na račun platforme Linux. Prodaja osrednjih računalnikov bo ostala stabilna.

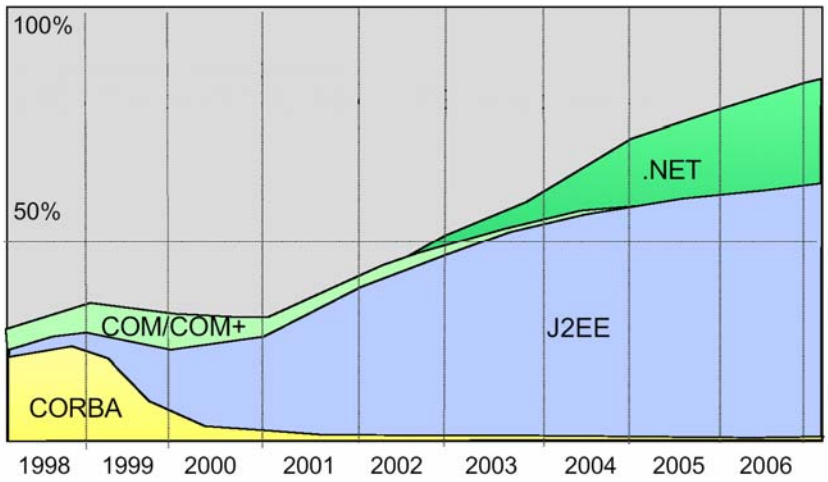
Slika 28: Predvidena prodaja platform (v milijonih dolarjev)



Vir: IDC Worldwide, 2004

Po raziskavah Gartner Group (Yefim Natis, 2003) bo do leta 2007 več kot 75% podjetij uporabljalo Microsoft in J2EE tehnologijo (verjetnost 0,8). Več kot 95% uporabnikov Windows platforme se bo odločilo za .NET. Za večino podjetij bo pomembnejše od tega katero tehnologijo izbrati, kako obe tehnologiji med seboj učinkovito povezati.

Slika 29: Predvideni delež posameznih tehnologij



Vir: Gartner Group, 2003

Porazdeljeno računalniško omrežje

Podjetja se vse bolj povezujejo regionalno in tudi globalno, zato postaja vse bolj zanimivo porazdeljeno računalniško omrežje. Prednost je v boljši izkoriščenosti omrežja, večji zanesljivosti (če posamezni računalnik v omrežju odpove, ga lahko nadomesti drugi), omrežje je bolj prožno in prilagodljivo. Poleg porazdeljenih omrežij obstajajo tudi porazdeljene baze podatkov. V tem primeru se podatkovna baza nahaja na več mestih. Baza podatkov je lahko porazdeljena tudi, če se na več mestih nahajajo kopije podatkovne baze. V današnjem času so najbolj pogosto uporablja kombinacija obeh primerov. V prihodnosti lahko pričakujemo, da se bodo hitrosti na komunikacijskih poteh povečale in zaradi tega bo v uporabi vse bolj slednji primer.

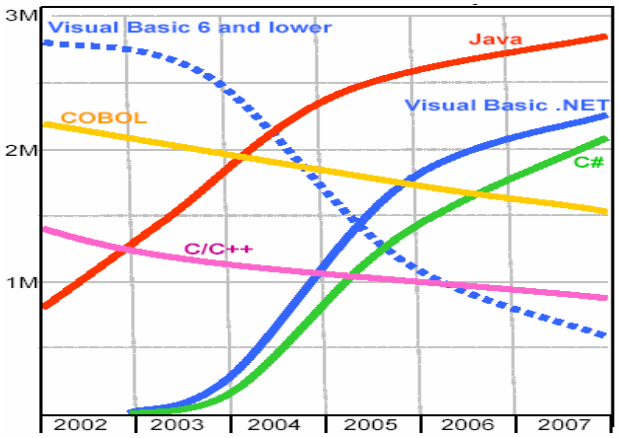
Trend programskih jezikov

V splošnem lahko rečemo, da gre razvoj jezikov v isto smer kot razvoj ostale informacijske tehnologije, in sicer v smeri čim bolj enostavne uporabe. Pomemben je hiter razvoj novih poslovnih aplikacij. V literaturi se omenjajo že programski jeziki pete in celo šeste generacije. Vsaka naslednja generacija jezika omogoča enostavnejše in bolj učinkovito programiranje. Na področju programskih jezikov je Microsoft naredil velik napredek z razvojem C#.

Raziskava Garner Group (Slika 30) ugotavlja, da se bodo v naslednjih letih številni razvijalci v C++ odločili za C#, razvijalci v Visual Basic 6.0 pa za Visual Basic .NET. Če sklepamo po tej

raziskavi, ima .NET lepo prihodnost, saj bo skupno število razvijalcev (Visual Basic .NET in C#) že v letu 2005 preseglo število razvijalcev v Javi.

Slika 30: Predvideno število razvijalcev za posamezne jezike (v milijonih)



Vir: http://www.asconline.com/pdfs/GartnerDotNETvsJavaSlides.pdf

7. SKLEP

Elektronsko poslovanje omogoča nove poslovne priložnosti, ki pa ponavadi zahtevajo temeljite spremembe v organizaciji, poslovnih procesih, načinu vodenja in kulturi podjetja. Pri zagotavljanju konkurenčnosti v spremenjenih pogojih poslovanja, ki jih prinaša nova poslovna paradigma, je potrebna tudi učinkovita uporaba informacijskih sistemov.

Zelo pomembno tehnologijo v elektronskem poslovanju predstavlja spletna storitev, ki je podobna storitvi v resničnem, fizičnem svetu. Spletne storitve so zasnovane tako, da se integrirajo v že obstoječe programske rešitve. Storitev je lahko brezplačna, namenjena javnosti za plačilo ali pa je preprosto mišljena kot storitev za partnerska podjetja. Ideja, ponuditi rešitev kot storitev, ki jo je mogoče dostaviti kadar koli, kamor koli in v katero koli napravo, je privlačna.

Moj osnovni avtorski prispevek je v analizi obeh najpomembnejših tehnologij .NET in J2EE. V raziskavi sem ugotovil podobnosti in razlike, prednosti in slabosti ene in druge tehnologije. Raziskal sem trenutno stanje in trende, ki jih lahko pričakujemo v informatiki. Med tema dvema tehnologijama je veliko podobnosti in tudi različnosti. Obe predstavljata zelo veliki tehnološki napredek. Preprosti odgovor na vprašanje katera tehnologija je boljša je nesmiseln brez analiziranja vseh pomembnih kriterijev. Tehnične zmogljivosti lahko primerjamo z različnimi kombinacijami ostale programske in strojne opreme, in za obe tehnologiji lahko

prikažemo, da je ena boljša, hitrejša itd. od druge. Pomembnejša je ugotovitev, da imamo dva zmagovalca v areni.

Izbira med J2EE in .NET ne sme temeljiti samo na tehnoloških razlikah. Zelo pomembno je, da odločitev predstavlja strateško poslovno odločitev platforme. Seveda morajo pri tem biti analizirane številne tehnične karakteristike. Pred nekaj leti je bilo očitno, da omogoča platforma J2EE veliko zmanjšanje stroškov TCO v primerjavi z ostalimi rešitvami. S pojavom .NET arhitekture pa je potrebno ponovno preučiti že sprejete odločitve.

Cilj magistrske naloge je dosežen. Z raziskavo sem ugotovil najpomembnejše kriterije, ki jih je potrebno upoštevati pri izbiri omenjenih tehnologij. Ugotovil sem, da odločitev ni preprosta in zgolj primerjava tehničnih karakteristik ni dovolj za smotrno odločitev. Organizacija mora najprej analizirati trenutno stanje informacijske podpore. Preučiti mora s kakšnimi kadri razpolaga. Na končno odločitev mora vplivati tudi razmislek o trendu razvijanja informacijskih rešitev.

Ocenjujem, da je .NET tehnologija boljša, še posebno če uporabljamo Windows platformo. .NET tehnologija je modernejša in v celoti objektna. Ponuja razvoj programskih rešitev naslednje generacije. XML je v .NET-u že v osnovi načrtovan. Povezljivost med različnimi programskimi jeziki je odlična in enostavna. Jeziki uporabljajo enak zbirnik, knjižnico razredov, itd. Koda jezika C# je krajša od Jave, število programerjev v C# in Visual Basic .NET-u se skokovito povečuje. Razvijanje spletnih storitev v .NET-u je enostavnejše in bolje integrirano. Zelo pomembno je tudi to, da .NET omogoča zelo produktivno razvijanje aplikacij, kar zmanjšuje stroške TCO. Prepričan sem, da se bodo manjša in srednje velika podjetja praviloma odločala za .NET tehnologijo, ker v teh podjetjih ponavadi temeljijo aplikacije na Windows platformi in COM+.

Nasprotno pa je J2EE izbira primernejša za večja podjetja, ki uporabljajo koncept velikih centralnih računalnikov. V teh podjetjih je že sedaj veliko strokovnjakov za Javo. J2EE omogoča boljšo povezljivost z obstoječo programsko opremo in različnimi aplikacijskimi strežniki . Pričakujemo lahko, da se bo v teh podjetjih pokazala potreba po uporabi obeh platform. Lahko bi rekli, da se zgodovina ponavlja tudi v informacijski tehnologiji, saj postaja koncept velikih osrednjih računalnikov ponovno aktualen.

8. LITERATURA

1. Abrian Turtschi, Jason Werry, Greg Hack, Joseph Albahari, C#.NET Web Developer's Guide, Syngress, 2002.
2. Alex Homer: ASP.NET 1.1. Indianapolis, IN : Wiley Publishing, 2004.
3. Andrew G. Blank: TCP/IP JumpStart – Internet Protocol Basics, Second Edition, Sybex, 2003.
4. Avison, David E., Information systems development : methodologies, techniques and tools. McGraw-Hill, 1996.
5. Balena, Francesco: Programming Microsoft Visual Basic .NET. Microsoft Press, 2002.
6. Bart A, DePetrillo: Razumeti Microsoft.NET, 2002.
7. Basha, Jeelani S.: Professional Java Web Services. Wrox Press, 2002.
8. Boar Bernard H.: Strategic Thinking for Information Technology. New York: John Wiley & Sons, 1997.
9. Cerar Miro: Temelji ustavne ureditve, človekove pravice in temeljne svoboščine, gospodarska in socialna razmerja. Ljubljana: Ministrstvo za notranje zadeve RS, 2001, 72 str.
10. Dalvi, Dinar: Professional XML for .NET Developers. Birmingham, Wrox Press Ltd, 2001.
11. Drucker Peter, Managing in the Next Society. New York: TrumanTalley Books, 2002.
12. Gradišar Miro: Uvod v informatiko. Ljubljana: Ekonomska fakulteta, 2003.
13. Jurič J. Matjaž, Jeelani Basha, RickLeander, Ramesh Nagappan: Professional J2EE EAI. Wrox Press, 2001.
14. Harold, Elliote Rusty: XML Bible. Foster City: IDG Books Worldwide, 1999.
15. Harold, Elliotte Rusty: XML Bible (2nd edition). New York: Hungry Minds, Inc., 2001.
16. Jerman Blažič Borka: Elektronsko poslovanje na Internetu. Ljubljana: GV založba, 2001.
17. Kovačič A., Groznik A., Indihar Štemberger M., Jaklič J.: Strateško načrtovanje poslovne informatike v slovenskih organizacijah. Uporabna informatika, Ljubljana, 2000.
18. Li Gong, Gary Ellison, Mary Dageforte: Inside Java 2 Platform Security, Second Edition, 2002..
19. Gradišar Miro in Resinovič Gortan: Informatika v organizaciji. Kranj: Moderna organizacija, 1998.
20. Gunnar Peterson: Component Security Design Considerations for J2EE and Net – An Architectural View, 2002.
21. Hiroshi Maruyama, Kent Tamura, Naohiko Uramoto, Makoto Murata, Andy Clark, Yuichi Nakamura, Ryo Neyama, Kazuya Kosaka, Satoshi Hada: XML and Java: Developing Web Applications, Second Edition, Addison Wesley, 2002.
22. Harold, Elliotte Rusty: XML Bible (2nd edition). New York: Hungry Minds, Inc., 2001. 1206 str.
23. Harumi Kuno: My Agent Wants to Talk to Your Service: Personalizing Web Services through Agents, Hewlett Packard Laboratories, 2002.
24. Kalakota Ravi, Robinson Marcia: e-Business Roadmap for Sucess, Addison- Wesley, 2001.
25. Kovačič Andrej in Groznik Aleš: Skladnost poslovnega strateškega načrta s strateškim načrtom informatike. Uporabna informatika, Ljubljana, 2001.
26. Kovačič Andrej: Informatizacija poslovanja. Ljubljana: Ekonomska fakulteta v Ljubljani, 1998.

27. Man Joung Rhee: Internet Security, Cryptographic principles, algorithms and protocols, Wiley, 2003.
28. Microsoft .NET vs. Sun Microsystem's J2EETM: The Nile Ecommerce Application Server Benchmark, 2001.
29. Michael Stoeckert: J2EE vs. Net. A Guide to Finding the Best Fit for your Institution, Version 62, 2002.
30. Mesbah Ahmed, Chris Garrett, Jeremy Faircloth, Chris Playne, ASP.NET Web Developer's Guide, Syngress, 2002.
31. Petroutsos, Evangelos: Mastering Visual Basic.NET, SYBEX, cop., 2002.
32. Reponen T: Strategic Information Systems – A Conceptual Analaysis. Journal of Strategic Information Systems, New York, 2003.
33. Roman Ed: Mastering EJB. New York: Wiley, 1999, 705 str.
34. Rob Keefer: SDS Strategic Data system, Ohio, 2002.
35. Scott Short: Building XML Web Services for the Microsoft .NET platform. Microsoft Press, 2002.
36. Stephen Walter: ASP.NET Unleased, Second edition. Indianapolis, Sams, 2004.
37. Strateško načrtovanje razvoja informatike. Doktorska disertacija, Ljubljana: Ekonomska fakulteta, 2001.
38. Tim H. Wong, Ganz Chokalingam: Enhancing Human and Web Interaction through Mobile Technology .NET vs. J2EE, Syngress, 2003.
39. Turban E. et al: Information technology for management - Making Connections for Strategic Advantage. New York: John Wiley & Sons. Inc., 1999.
40. Thuan Thai, Hoang Q. Lam, .NET Framework Essential, O'Reilly, 2001.
41. The Middleware Company Case Study Team: J2EE and .NET (RELOADED) Yet Another Performance Case Study, 2003.

9. VIRI

1. Bourret Ronald: XML and Databases. [URL:http://www.rpbourret.com/xml/XMLAndDatabases.htm], 19.11.2003.
2. Deshmukh Rajan: Understanding the Business Models for the Internet Economy, [URL:http://pegasus.rutgers.edu/~rajadesh/Project_reports.htm], 2000.
3. Durchslag Scott et al.: Web Services: Enabling The Collaborative Enterprise. [URL:http://eserv. ebizq.ner/wbs/donato_1.pdf], 01.07.2001.
4. Dustin Marx: More JSP best practices. [URL: http://www.javaworld.com/javaworld/jw-07-2003/jw-0725-morejsp.html?], 2003.
5. E-Business Manufacturing. Camstar Systems, Inc.. [URL:http://b2b.ebizq.net/shared/white_papers.jsp?ID=cone_1.pdf], 2001
6. Implementing Sun Microsystem's JavaTM Pet Store J2EETM Blueprint Application using Microsoft .NET [http://www.gotdotnet.com/compare].
7. Java 2 Platform Enterprise Edition Blueprints website: [URL: http://java.sun.com/j2ee/blueprints/index.html].

8. Jonathan Lurie: The great debate: .NET vs. J2EE.
[URL: http://www.javaworld.com/javaworld/jw-03-2002/jw-0308-j2eenet.html?], 08.03.2002.
9. Jason Hunter: Servlet 2.3: New features exposed.
[http://www.javaworld.com/javaworld/jw-01-2001/jw-0126-servletapi.html?], 2002.
10. James Kao: developer's Guide to Building XML – based Web Services.
[http://www.theserverside.com/articles/article.tss?l=WebServices-Dev-Guide], 2001.
11. Mark Johnson: A walking tour of JavaBeans.
[URL: www.javaworld.com/jw-08-1997/jw-08-beans-p1.html]
12. Matt Berger: Sun Adds Web Services to J2EE.
[URL: http://www.javaworld.com/javaworld/jw-01-2002/jw-0125-webservices.html], 2001.
13. Michael Morrison: Presenting JavaBeans.
[URL: http://docs.rinet.ru/JavaBeans/html/fm.htm]
14. RIS: Podjetja z dostopom do interneta. Fakulteta za družbene vede, Ljubljana.
[URL:http://www.ris.org/si/ris99/podjetja1.htm], 12.09.2002.
15. World Wide Web Consortium. Web Style Sheets. [URL:
http://www.w3.org/TR/1998/REC-xml-19980210http://www.w3.org/Style/].
16. Wong Sam: Web Services: The Next Evolution of Application Integration.
[URL: http://e-serv.ebizq.ner/wbs/wong_1.html], 20.08.2001.
17. Sisto Joseph: Finding the right portal product, [URL:http://www.northhighland.
com/whitepaper_Sisto_portals_2.htm], 24.11.2002. 3 str.
18. Zakon o elektronskem poslovanju in elektronskem podpisu (ZEPEP). (Uradni list RS, št. 57/2000).