SMALLTALK NA INTERNETU

Janko Mivšek

TRIS A d.o.o., Pod hribom 55, Ljubljana

tel.: (061) 159-54-39

e-pošta: janko@tris-a.si

URL: http://www.tris-a.si

Povzetek

S tem člankom želimo predstaviti naše dosedanje in bodoče delo pri uvajanju objektne tehnologije za podporo poslovanju na Internetu. Na konkretnem projektu Internet prodaje razvijamo podporno tehnologijo, ki nam bo omogočala dinamične predstavitve ter tesno integracijo Internet storitev z objektno naravnanimi informacijskimi sistemi. Sistem razvijamo v objektnem jeziku Smalltalk in na objektni bazi podatkov. V članku skušamo prikazati filozofijo našega pristopa, za katerega menimo, da je izviren glede na podobne napore v svetu. Opisujemo tudi razloge za izbiro objektne tehnologije ter načrte za nadaljnjo organizacijo našega objektno naravnanega razvojnega procesa.

1. Uvod

Internet prodira z veliko hitrostjo v poslovno življenje podjetij, saj se odpirajo možnosti poslovanja s 50 miljoni potencialnih kupcev po celem svetu. Pri nas je ta številka po zadnji raziskavi [9] ocenjena na 50.000 do 70.000 in skokovito narašča. Vendar, kako poslovni informacijski sistem odpreti tako, da bo dostopen uporabnikom z Interneta? Sedanji način priprave informacij v obliki 'domačih strani' je preneroden za vzdrževanje in spreminjanje, zato potrebujemo način, s katerim bomo pripravljali in obnavljali predstavitev enostavno in čimbolj samodejno.

Objektna tehnologija obljublja ravno to - hiter razvoj, prilagodljivost in enostavno vzdrževanje. Zato smo se odločili, da združimo to tehnologijo z Internetom in s tem odpremo nove, mnogo širše možnosti poslovanja ob uporabi tega novega medija.

Namen članka je predstavili konkreten projekt sistema kataloške prodaje preko Interneta. Sistem zajema izbiro artiklov iz kataloga, naročanje in dostavo. Načrtujemo ga z uporabo objektnih metod (Jacobson), implementiramo v programskem jeziku Smalltalk (ParcPlace VisualWorks) in objekte shranjujemo v objektni bazi Versant.

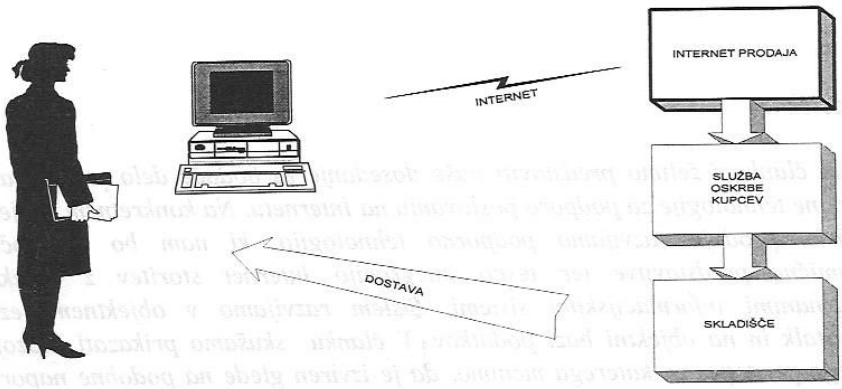
2. Poslovanje na Internetu

S širjenjem Interneta so se odprle nove možnosti poslovanja podjetij, ker ta lažje pridejo do svojih kupcev. Na svetovnem spletu (World Wide Web - WWW, v nadaljevanju kar Internet) lahko kot kupec iščemo, izbiramo in naročamo preko svojega računalnika neposredno pri dobavitelju kjerkoli na svetu. Odpade mučno zbiranje informacij po trgovinah, odpade

vsiljivost raznih trgovskih potnikov in prodajnih katalogov po pošti. Preko Internetu naročamo, kadar in kar si mi zaželimo.

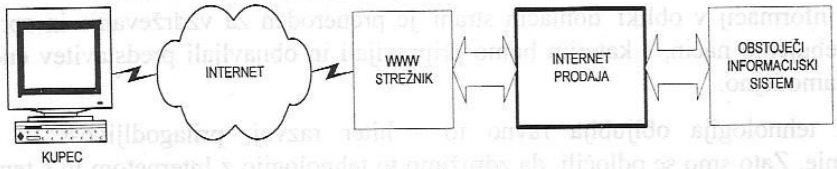
Največ poslovanja zavzema prodaja na Internetu, ki v grobem poteka v naslednjih korakih:

1. kupec na svojem računalniku izbira iz kataloga pri dobavitelju,
2. kupec izbrano blago označi in sestavi ter pošlje elektronsko naročilnico,
3. dobaviteljeva služba oskrbe kupcev sprejme naročilo in izda zahtevek za dostavo skladišču,
4. skladišče dostavi blago kupcu na dom, pošlje po pošti, ipd.,
5. kupec plača blago po povzetju, s kreditno kartico ali po izstavljenem računu.

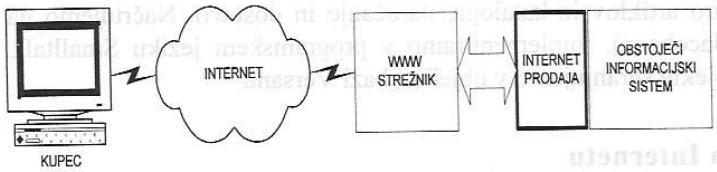


slika 1. Potek Internet prodaje

Ker je večina zgornjih korakov že podprta z obstoječim informacijskim sistemom, je zaželejno, da Internet prodajo nekako vkjučimo v le-tega. Najboljši rezulat doseže popolna integracija, kjer postane Internet poslovanje le del celote in ne poseben sistem, ki je z vmesniki povezan z osrednjim sistemom.



a. Ločena Internet prodaja

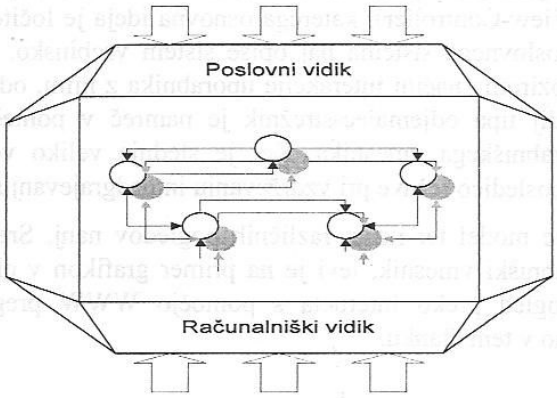


b. Internet prodaja, integrirana v informacijski sistem

slika 2. Ločen ali integriran sistem Internet prodaje

3. Razvojni proces

Naša filozofija razvoja objektno naravnanih informacijskih sistemov se naslanja na deli Taylorja [17] ter Jacobsona [9], ki predlagata združitev snovanja informacijskih sistemov s snovanjem organizacije poslovnega sistema podjetja. Osnova za skupno snovanje pa je razumevanje med informatiki in organizatorji poslovnega sistema, torej nekakšen skupen jezik. Oba avtorja zagovarjata objektno modeliranje kot tehniko komuniciranja, ki je dovolj naravna in blizu organizatorjem, da jo lahko vzamejo za svojo.



slika 3. Enoten pogled na model poslovnega sistema

Model je torej osnova za gradnjo poslovnega sistema tako s poslovnega vidika kot z računalniškega vidika (slika 3).

Model lahko uporabljamo na tri različne načine:

1. *Za predstavitev* strukture in delovanja poslovnega sistema na enoten način, tako, da poslovanja lažje razumemo in najdemo nove poti za njegovo izboljšanje;
2. *Za simulacijo*: model lahko uporabimo za prognoziranje delovanja sistema v bodoče, kot

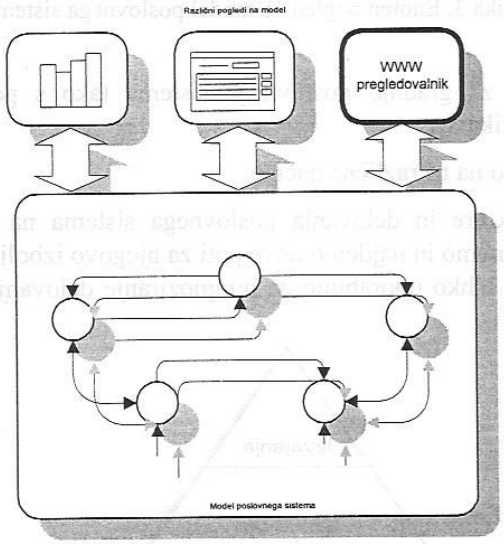
slika 4. Tri funkcije modela poslovnega sistema

na primer za prognozo finačnih tokov. Simulacija je tudi močno orodje za analizo posledic organizacijskih sprememb (What-If analiza) ter za iskanje optimalnega načina poslovanja;

3. Za *izvajanje:* model naj bo sposoben tudi izvajati poslovna opravila ter s tem razbremeniti podjetje rutinskih opravil (ta funkcija je sicer značilna za klasične informacijske sisteme). Izvajanje modela ne pripomore samo pri avtomatizaciji poslovnih opravil, ampak tudi zagotovi, da se opravila izvajajo v skladu z modelom.

Pri izgradnji naših sistemov želimo ohraniti čimveč dosedanjih izkušenj iz Smalltalk okolij, predvsem pri izgradnji grafičnih uporabniških vmesnikov. Pri snovanju le-teh se je uveljavil model MVC (Model-View-Controller), katerega osnovna ideja je ločitev modela od njegove predstavitve. Model poslovnega sistema naj opiše sistem vsebinsko, medtem ko naj bodo različni pogledi nanj, oziroma načini interakcije uporabnika z njim, od njega ločeni. Tipičen problem sedanjih orodij tipa odjemalec-strežnik je namreč v pomešanosti fukcionalnosti modela s kodo uporabniškega vmesnika. Ker je slednje veliko več, se funkcionalnost porazgubi, kar ima za posledico težave pri vzdrževanju in nadgrajevanju sistema.

Spodnja slika prikazuje model ter nekaj različnih pogledov nanj. Srednji pogled prikazuje klasični grafični uporabniški vmesnik, levi je na primer grafikon v elektronski preglednici, desni pa prikazuje pogled preko Interneta s pomočjo WWW pregledovalnika. Izvedbo zadnjega tudi opisujemo v tem članku.



slika 5. Model ter različni pogledi nanj

Poudariti je še treba, da lahko med 'poglede' štejemo tudi vmesnike do drugih sistemov, kot je na primer vmesnik za elektronsko izmenjavo podatkov (EDI) z našimi partnerji.

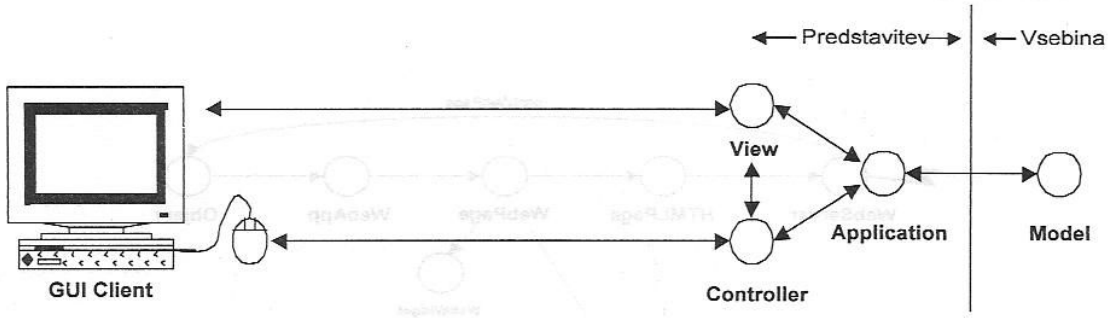
Naj še enkrat poudarim ločitev modela od pogledov nanj. Model naj se ne ukvarja s predstavitvijo samega sebe, zanjo naj skrbijo različne vrste pogledov. Model se torej ne zaveda prisotnosti pogledov, medtem ko se pogledi modela zavedajo. S tem dosežemo večjo stabilnost sistema glede sprememb, ker praksa kaže, da je največ sprememb potrebno na

uporabniških vmesnikih in mnogo manj na vsebini modela. Tako dosežemo, da je sistem pregleden in ga je lažje vzdrževati.

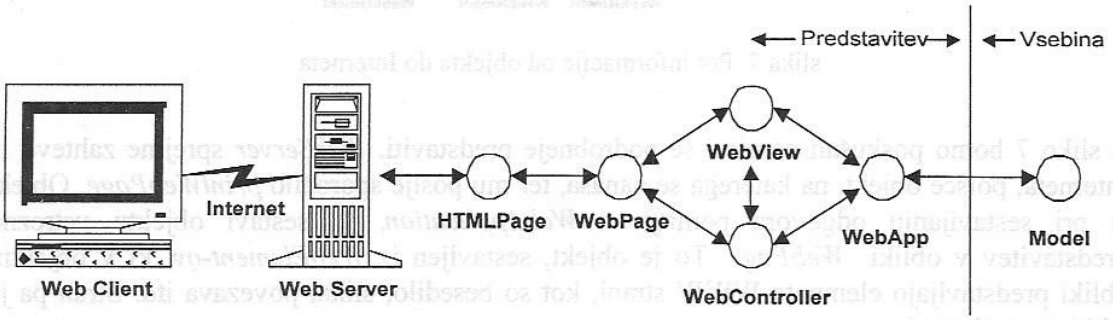
Povezavo informacijskega sistema v Internet torej najlažje dosežemo tako, da naredimo nove poglede na naš model, posebej prirejene za prikazovanje informacij na Internetu.

4. Povezava informacijskega sistema z Internetom

Kot smo videli na sliki 5, je smiselno informacijski sistem priključiti v Internet tako, da ustvarimo novo vrsto pogledov, posebej prilagojenih za prikaz informacije in interakcijo v WWW pregledovalnikih. Omenili smo že arhitekturo MVC, ki jo bomo prilagodili za delo na Internetu.



a. Smalltalk MVC Arhitektura



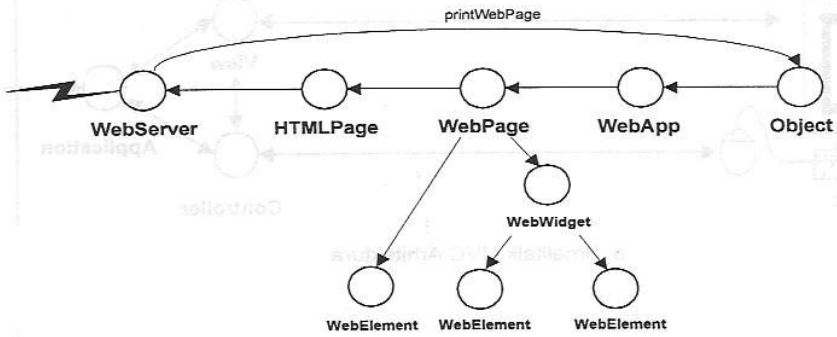
b. Internet MVC Arhitektura

slika 6. Primerjava Smalltalk in Internet MVC arhitektur

Na sliki 6 vidimo tipičen izgled sistema pri uporabi grafičnega uporabniškega vmesnika. Na model, ki predstavlja vsebinsko plat sistema, gledamo s pogledom (View) , ki prikaže enega od vidikov modela na nam primeren način (grafično, tekstovno ipd.). Vzporedno s pogledom pa imamo še krmilnik (Controller), ki prestreže našo interakcijo s pogledom, npr. (klik z miško na gumb v oknu) ter pripravi ustrezno sporočilo za model (npr. shrani vpisano vsebino v oknu). Na tak način smo dosegli, da smo del programske opreme, ki skrbi za predstavitev in interakcijo s sistemom (te je ponavadi okoli 80%) ločili od vsebinskega dela sistema, predstavljenega v modelu.

V drugi polovici slike 6 predstavljamo naš predlog prilagoditve MVC arhitekture za Internet. Našo zamisel o izgradnji WWW pogledov lahko strnemo v naslednjih točkah:

- informacijo predstavimo v standardni HTML obliki (HTMLPage),
- pred HTML obliko predstavimo stran (WebPage) kot skupek objektov (stran, besedilo, odstavek, povezava, slika, itd..) za lažjo obdelavo v objektnem sistemu,
- pri sestavljanju strani pomaga posebna aplikacija (WebApplication), poleg tega se pa ustrezno odziva na uporabnikove zahteve (kot npr, poslana izpoljnjena naročilnica)
- omogočimo, da se vsak objekt v sistemu zna predstaviti kot WWW stran. Pri tem mu pomaga aplikacija
- izdelamo 'pametne' komponente WWW strani (kot npr. razni seznami, ikonske vrstice za navigacijo, ipd).



slika 7. Pot informacije od objekta do Interneta

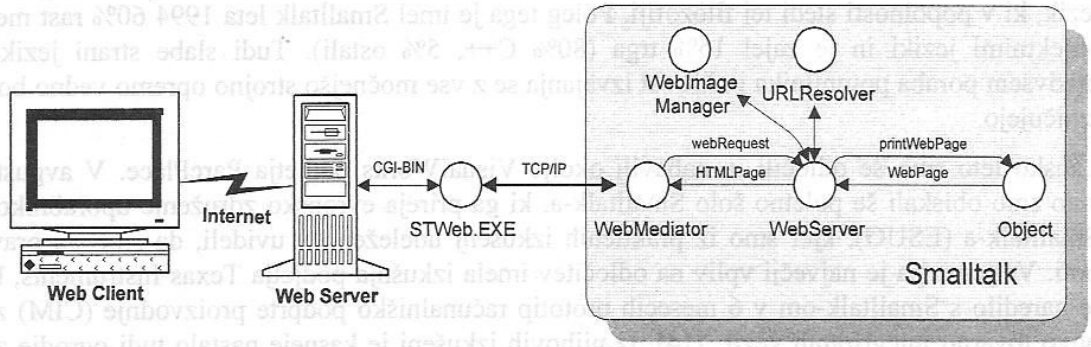
Z sliko 7 bomo poskušali zamisel še podrobneje predstaviti. *WebServer* sprejme zahtevo iz Interneta, poišče objekt, na katerega se nanaša, ter mu pošlje sporočilo *printWebPage*. Objekt si pri sestavljanju odgovora pomaga z *WebApplication,* ki sestavi objektu ustrezno predstavitev v obliki *WebPage.* To je objekt, sestavljen iz *WebElement-ov,* ki v objektni obliki predstavljajo elemente WWW strani, kot so besedilo, slika, povezava itd. Stran pa je lahko sestavljena iz

WebWidget-ov. Medtem, ko so *WebElement-i* statični gradniki strani, prinašajo *WebWidget-i* dinamičnost. Ti gradniki se znajo sami sestaviti glede na vhodne podatke. Lep primer je na primer seznam zadetkov na iskalnikih, kot so Lycos ali AltaVista. Vhodni podatek je seznam zadetkov, kot izhod pa dobimo urejeno HTML obliko seznama. Naslednji primer je pas ikon na Web straneh, ki služijo za navigacijo. Tudi standardno ogrodnje za prazno WWW stran lahko sestavimo v obliki *WebWidget-a. WebWidget*-e lahko tudi podedujemo. Na našem podjetju ravnokar sestavljamo hierarhijo *WebWidget*-ov za standardne dele WWW strani.

Na koncu še poglejmo, kako naša zamisel izgleda v realnosti. Na Internetu si lahko pogledate eksperimentalni Smalltalk WWW strežnik v na naslednji strani: *http://www.tris-a.si/stweb/.* Na sliki 8 prikazujemo delovanje tega strežnika.
Uporabnik zahteva preko interneta določeno WWW stran. Iz formata URL (Uniform-Resource-Locator) zahteve WWW strežnik (npr. Netscape Server) ugotovi, da mora preko CGI-BIN vmesnika poslati zahtevo v Smalltalk sistem. Primer URL formata je naslednji:

```
http://www.tris-a.si/cgi-bin/stweb?/demo/overview.html
```

Za vsak klic v Smalltalk sistem moramo torej poklicati CGI-BIN program STWeb.exe, ki prenese parametre klica preko TCP/IP protokola v Smalltalk sistem. Tu ga sprejme *WebMediator,* ki deluje kot posrednik med klasičnim WWW strežnikom ter Smalltalk sistemom. Med drugim sestavi tudi sporočilo *webRequest* in ga pošlje objektu *WebServer.* Ta ima nalogo, da na osnovi URL najde referenco na pravi objekt, ki je lahko običajen objekt, objektna WWW stran (*WebPage*) ali HTML stran (*HTMLPage*), skratka, vsak objekt se mora znati predstaviti kot WWW stran.



slika 8. Vključitev Smalltalk sistema v Internet

Pri pretvorbi iz URL v objektno referenco pomaga WebServer-ju URLResolver, ki hrani tabelo povezav URL- objekti. Tudi objektne WWW strani imajo namreč povezave na druge objektne strani določene v obliki objektnih referenc in ne kot URL, kar zopet pomaga pri lažjem delu z WWW informacijo v objektnem sistemu.
Ciljni objekt mora torej sestaviti objekt *WebPage*. Če sam tega ne zna, si lahko pomaga s posebnim objektom, ki ga poimenujmo WWW aplikacija (*WebApplication*). Aplikacija pride posebej prav pri kompleksnejših predstavitvah, nujna pa je takrat, ko WWW klic vsebuje tudi povratne informacije iz WWW obrazcev. Tipičen primer je naročanje, kjer izpolnimo naročilnico in jo pošljemo. Objektna stran se nato pretvori v HTML stran, ki jo *WebServer* preko *WebMediator-ja* pošlje nazaj k uporabniku.

Čeprav izgleda na prvi pogled postopek zapleten, nam omogoča, da ločimo posebnosti WWW predstavitve in prikaz informacij na Internetu izvedemo na objekten način, tako kot je zgrajen ostali del sistema.

5. Zakaj Smalltalk, zakaj objektna baza?

Zgodba se začenja leta 1994, ko smo začeli iskati zamenjavo za dotedanje razvojno orodje, ki smo ga naredili sami na Inštitutu Jožef Štefan in je nudilo dovolj možnosti za izgradnjo dokaj popolnih grafičnih uporabniških vmesnikov pod OS/2 ter povezavo z relacijsko bazo DB2. Vendar pa je bil razvoj s tem orodjem prepočasen, narejeno programsko opremo je bilo težko vzdrževati. Zato smo se lotili iskanja novega orodja, ki naj bi pokril naše potrebe po hitrejšem razvoju tudi kompleksnejših sistemov, lažjem vzdrževanju ter naj bi vzdržal vsaj 5-10 let. Na izbiro smo imeli orodja tipa odjemalec-strežnik, kot so PowerBuilder, Gupta SQLWindows ipd, vendar nas nobeno ni zadovoljilo, ker so bila vsa preveč zaprta v Windows okolje, s

prevelikim poudarkom na izgradnji uporabniških vmesnikov ter samo na relacijskih bazah podatkov.

Leta 1994 je dal IBM na trg VisualAge, to je razvojno okolje, ki temelji na jeziku Smalltalk. Pri podrobnejšem preučevanju tega in drugih Smalltalk orodij ter študiju referenčnih projektov smo ugotovili, da je Smalltalk zelo resno orodje in da se v tej smeri splača razmisliti. Objektna tehnologija je takrat šele prehajala iz modne v resnejšo sfero, tako da smo bili do nje rahlo nezaupljivi. Vendar smo po temeljitejšem študiju literature ugotovili, da ima ta tehnologija dovolj jasno in trdno filozofijo, ki je nastajala že v 60-tih letih in torej ni muha enodnevnica. Poleg tega smo spoznali, da je pravzaprav Smalltalk edini komercialni objektni jezik, ki v popolnosti sledi tej filozofiji. Poleg tega je imel Smalltalk leta 1994 60% rast med objektnimi jeziki in je zajel 15% trga (80% C++, 5% ostali). Tudi slabe strani jezika, predvsem poraba pomnilnika in hitrost izvajanja se z vse močnejšo strojno opremo vedno bolj izničujejo.

Lansko leto smo se odločili in nabavili okolje VisualWorks podjetja ParcPlace. V avgustu smo smo obiskali še poletno šolo Smalltalk-a, ki ga prireja evropsko združenje uporabnikov Smalltalk-a (ESUG), kjer smo iz praktičnih izkušenj udeležencev uvideli, da smo na pravi poti. Verjetno pa je največji vpliv na odločitev imela izkušnja podjetja Texas Instruments, ki je naredilo s Smalltalk-om v 6 mesecih prototip računalniško podprte proizvodnje (CIM) za novo tovarno integriranih vezij [14]. Iz njihovih izkušenj je kasneje nastalo tudi ogrodje za objektno naravnane proizvodne informacijske sisteme [2], ki je predloženo OMG v izbiro za vertikalno domeno prozvodnje (glej tudi sliko 9).

Pred nami je ostala še dilema, ali vztrajati pri relacijskih ali preiti na objektne baze podatkov. Relacijske baze so dodobra utečene ter robustne in tudi vsa orodja jih podpirajo. Vendar smo po nekaj mesecih uporabe Smalltalk-a začutili, da bomo imeli z relacijskimi bazami preveč težav pri preslikavi objektov v tabele in obratno. Svoboda modeliranja v Smalltalk-u je zadela ob zid dvodimenzionalnih podatkov v relacijskih bazah. Poleg tega hitrost relacijskih baz pri kompleksnejših modelih hitro pada. Zato smo se začeli spogledovati z objektnimi bazami.

Začetni študij je pokazal, da lahko računamo na zelo dobro integracijo baze v jezik Smalltalk ter na ugodno hitrost (10-100x hitrejše, [3]), kar gre pripisati navigacijskemu načinu dostopa s kazalci nasproti samo asociativnim povezavam v relacijskih bazah. Skrbela nas je le razširljivost in robustnost objektnih baz. Ker pa so se objektne baze najbolj uveljavile ravno v telekomunikacijski industriji, kjer je prvi pogoj pri snovanju zanesljivost in robustnost sistemov, je bilo očitno da so te baze na tem področju že dovolj zrele. Kar se razširljivosti tiče, dosegajo produkcijske baze že 100Gb, experimentalno pa celo 400Gb, kar je za večino primerov dovolj.

Trenutno so najbolj razširjene naslednje objektne baze: ObjectStore, Versant, GemStone, Objectivity, O2, Poet. Vse imajo povezave na jezik C++, večina tudi na Smalltalk. Najprej smo dobili v test ObjectStore [13], vendar so Smalltalk vmesnik šele gradili, pa tudi zanesljivost ter porazdelitveni model sta bila vprašljiva. GemStone ni hotel sodelovati z nami (menda so se bali zaradi zaščite avtorskih pravic), zato smo povprašali še Versant. Versant že od začetka podpira tako C++ kot Smalltalk [19]. Ima tudi zelo dober porazdelitveni model, saj lahko poljubno porazdelimo do 64000 baz in med njimi med delovanjem premikamo objekte iz ene baze v drugo. Integracija v Smalltalk je tako dobra, da skoraj ne opazimo, da je naš objekt pravzaprav zapisan v bazi. Objektne baze namreč samodejno shranijo vsak objekt, če nanj pokažemo iz že shranjenega objekta. Če pa želimo shraniti svež objekt, mu pošljemo

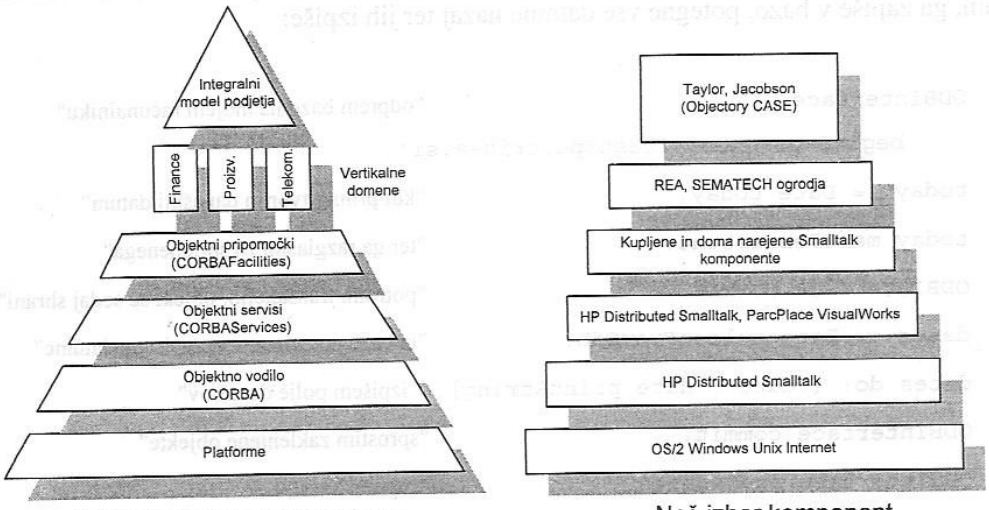
sporočilo *makePersistent*. Naj delo z objektno bazo ponazorim s primerom, ki tvori današnji datum, ga zapiše v bazo, potegne vse datume nazaj ter jih izpiše:

```
ODBInterface                              "odprem bazo na mojem računalniku"
    beginSessionOn: 'test@pu.tris-a.si'.
today := Date today.                      "kot primer tvorim današnji datum"
today makePersistent.                     "ter ga razglasim za shranjenega"
ODBInterface commit.                      "potrdim transakcijo, objekt se sedaj shrani"
dates := Date selectFromOdb.              "iz baze potegnem vse zapisane datume"
dates do: [:date | date printString].     "izpišem polje datumov"
ODBInterface commit.                      "sprostim zaklenjene objekte"
ODBInterface endSession.                  "zaprem bazo"
```

Trenutno še izbiramo razvojno metodologijo ter orodja za skupinsko delo, konfiguracijsko kontrolo ter kontrolo verzij. Nam najbližja se zdi Jacobsonova metodologija, ki je udejanjena v razvojnem procesu in orodju Objectory švedskega podjetja z enakim imenom. Razlogov za to je več:

- Objectory ni samo CASE, ampak tudi razvojni proces,
- orodje je zraslo na 25 letnih iskušnjah,
- Jacobson razširja metodologijo 'primerov uporabe' na podporo prenove poslovnih sistemov [9]
- orodje je tesno povezano z okoljem VisualWorks in omogoča sledljivost modela od zgoraj do Smalltalk-ovih objektov in nazaj.
- podjetje Objectory se je priključilo podjetju Rational, tako da razvijajo sedaj Jacobson, Rumbaugh in Booch enotno objektno metodologijo.

Na sliki 9 predstavljamo načrt uvajanja objektne tehnologije v našem podjetju v primerjavi s komponento arhitekturo OMG [2].



slika 9. Izbor objektne tehnologije v primerjavi z OMG komponentno arhitekturo

Po enem letu izkušenj lahko rečemo, da smo z odločitvijo zadovoljni, saj nam je odprla možnost gradnje kompleksnejših informacijskih sistemov. Imamo občutek, da smo se končno rešili stalnih problemov in omejitev programske opreme in da lahko svoje zamisli in ideje uresničimo hitreje, brez komplikacij računalniške narave. Sedaj nam ostane le, da prepričamo naše stranke, ki so preplavljene s ponudbo orodij različne kakovosti, v prednosti našega pristopa, tako da bomo lahko skupaj gradili boljše informacijske sisteme, ki bodo prinašali vidne in merljive prihranke v poslovanju.

7. Zaključek

Namen tega članka je bil v prvi meri ponazoriti moč objektne tehnologije na primeru uporabe Internet storitev v poslovanju podjetja. Iz dosedaj nabranih iskušenj smo prepričani, da meja tehnologije še zdavnaj nismo dosegli in da smo dobili v roke dovolj močno in obvladljivo orodje, da se lahko lotimo tudi kompleksnejših problemov. Spoznali pa smo tudi, da se bomo morali odslej posvečati bolj modeliranju, torej opisovanju bistva problema in manj sami implementaciji. Trenutno pa je naš prvi cilj postaviti temelje z izgradnjo dovolj močne objektne knjižnice ter ogrodij za izgradnjo poslovnih sistemov. V tem članku smo opisali samo delček tega, kar še želimo narediti.

8. Literatura

1. Carnegie Mellon University, Software Engineering Institiute: The Capability Maturity Model, Guidelines for Improving the Software Process, Addison Wesley, 1995

2. Cory Casanove: *Business Object Architectures and Standards,* OOPSLA 95

3. R.G.G Cattell: *Object Data Management, Revised Edition,* Addison Wesley, 1994

4. Tom Digre: *Business Object Facility,* OMG Business Object Architecture team, 1994

5. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns, Elements of Reusable Object-Oriented Software,* Addison Wesley, 1995

6. Guido L. Geerts, William E. McCarthy: *Modelling Business Enterprises as Value-Added Process Hierarchies with Resource-Event-Agent Object Templates,* Paper submited to OOPSLA95, 1995 (http://www.tiac.net/users/jsuth/oopsla/mcarpub.pdf)

7. Simon Lewis: *The Art and Science of Smalltalk*, Helwett Packard professional books, Prentice Hall, 1995

8. Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgaard: *Object-Oriented Software Engineering: A Use Case Driven Approach,* Addison.Wesley, 1992

9. Ivar Jacobson, Maria Ericsson, Agneta Jacobson: *The Object Advantage: Business Process Re-engineering with Object Technology*, Addison Wesley/ACM Press, 1995

10. Ministrstvo za znanost in tehnologijo, Fakulteta za družbene vede: *Raba Interneta v Sloveniji,* 1996 (http://www2.arnes.si/ris/)

11. Robert Orfali, Dan Harkey: *The Essential Distributed Objects Survival Guide*, John Wiley & Sons, 1996

12. Eddie Obeng, Stuard Crainer: *Making Re-engineering Happen,* Financial Times, Pitman Publishing, 1994

13. ObjectStore: *User Guide, ObjectStore Release 3.0 for OS/2 and AIX/x1c Systems,* Object Design, Inc., 1994

14. ParcPlace, Inc.: *Texas Instruments Builds Fab of the Future With Help From Objectworks Smalltalk,* Application Story, 1991

15. ParcPlace, Inc.: *VisualWorks User's Guide,* ParcPlace Systems, Inc., 1994

16. SEMATECH: *Implementation Handbook for the Computer Integrated Manufacturing (CIM) Application Framework Specification 1.2*, 1995, (http://www.sematech.org/public/cim-framework/2917aeng.pdf)

17. David A. Taylor: *Business Engineering with Object Technology*, John Wiley & Sons, 1995

18. David A. Taylor: *Object Oriented Information Systems: Planning and Implemenation*, John Wiley & Sons, 1992

19. Versant: *Smalltalk/VERSANT Manual,* Versant Object Technology, 1995