STRATEGIJA RAZVOJA INFORMACIJSKIH SISTEMOV ZA PODPORO ODLOCANJU

Franc Žerdin, Republiški sekretariat za pravosodje in upravo, Župančičeva 3, Ljubljana

1. UVOD

Informatizacija je postala dejstvo. Čeprav v razvoju niti približno ne sledimo razvitemu svetu, se procesu informatizacije ne moremo izogniti. Sodobna, uspešna podjetja bodo morala svojo organiziranost podrediti informacijski tehnologiji, ki bo omogočila avtomatizacijo poslovanja. Avtomatizacija poslovanja postaja imperativ, zlasti na zahtevnih mednarodnih trgih. Tako se bodo podjetja morala odpovedati togi hierarhični organizacijski strukturi in vpeljati novo strukturo, ki bo delovala podobno kot deluje simfonični orkester (P. Drucker)."Partitura" takega orkestralno organiziranega podjetja postane informacija, ki je shranjena v podatkovnih bazah podjetja. Podjetja, ki bodo svoje poslovanje osnovala na informacijah oz. znanju, bodo opustila togi birokratski model organiziranosti in bodo vpeljala fleksibilni model organiziranosti. Informacijsko naravnana podjetja pa bodo prišla do pravih informacij le,če bo zagotovljeno enotno planiranje in zasnova podatkov, oz. sodobna izgradnja računalniško zasnovanega integralnega informacijskega sistema. Ta služi tudi kot osnova za izdelavo sistemov za podporo odločanju v podjetju.

Gradnjo integralnih informacijskih sistemov omogočajo šele v zadnjem času dozorele razvojne metodologije,ki so razvoj sistemov premaknile od razvoja posameznih neodvisnih aplikacij ali delnih sistemov k razvoju planiranih, široko integriranih in strateško orientiranih sistemov. Tako zgrajeni sistemi nudijo popoln obseg informacij za celotno podjetje. Obdržale se bodo le tiste razvojne metodologije, ki zadoščajo nekaterim osnovnim

kriterijem kot so: modeliranje poslovnega sistema, strateško planiranje informacij, podatkovno in postopkovno modeliranje, avtomatizacija sistemske zasnove in izdelave programov, vključitev uporabnikov v zasnovo in izdelavo prototipov, še posebno pomembna pa je vključitev poslovodstva, da postavi prioritete in definira informacijske potrebe, še zlasti za področje podpore odločanju. Važni so še nekateri drugi aspekti. Tako mora metodologija zagotavljati uporabo avtomatskih orodij, ki pomagajo pri analizi in zasnovi, uporabo aplikacijskih generatorjev in jezikov 4. generacije. Važno je tudi, da metodologija zagotavlja dobro skrbništvo podatkov in pomoč informacijskim centrom, ki direktno pomagajo uporabnikom pri njihovem delu z osebnimi računalniki.

2. PROBLEMI RAZVOJA INFORMACIJSKIH SISTEMOV

Veliko je bilo povedanega in napisanega o tem, kaj je narobe z obdelavo podatkov. V večjih centrih imajo zlasti velike težave s proizvodnjo aplikativne programske opreme, ki pogosto zaostaja tudi več let za pričakovanji. Gradnja večjih sistemov traja predolgo. Stroški delujočih sistemov so izredno visoki, saj se profesionalne ekipe pogosto ukvarjajo le z vzdrževanjem obstoječih sistemov. Ker se razvijajo aplikacije praviloma brez sistemskega pristopa več ali manj neodvisno, je veliko redundantnih podatkov, ki jih različno ažurirajo. Zato ni redek pojav, da poslovodstvo ne more dobiti informacij iz računalnika, čeprav so v njem zbrani vsi potrebni podatki. Problemi pa so še večji, ko poslovanje zahteva naglo spremembo poslovnih postopkov, ali uvedbo novih proizvodov oz. storitev. Obdelava podatkov sploh ne sledi dovolj hitro takšnim spremembam in tako postane ovira ne pa podpora poslovnemu sistemu. Povedano drugače: razvoj aplikativne programske opreme ne more slediti dinamičnim, stalnim spremembam poslovnega sistema. Zato ni čudno, da je znani ameriški časopis "The Wall Street Journal" potožil: "Glavni zavori gospodarskega razvoja sta nafta in razvoj programske opreme!"

Našteti problemi se v glavnem nanašajo na centre z večletnimi izkušnjami.Povsod tam, kjer šele začenjajo z uvajanjem avtomatske obdelave podatkov, so problemi še hujši, saj delajo začetniške napake kot je razvoj neodvisnih aplikacij in s tem proizvajanje odvečnih podatkov, kar onemogoča kakršno koli kasnejšo integracijo sistemov. Razmere so težke zlasti tam, kjer uvajajo nenačrtno mikroračunalnike.

Ta zelo resen poslovni problem mora biti rešen. Kako? Ozrimo se na druga področja. Modernih letal, mikrorezin ("microchips") ali avtocest ne bi mogli graditi brez ustreznih avtomatskih naprav (avtomatskih orodij). Nekaj podobnega mora veljati pri gradnji računalniško zasnovanih informacijskih sistemov.Saj je nevzdržno, da se lotevamo zasnove poslovnih informacijskih sistemov, ki so enako kompleksni kot je zasnova mikrorezin ali reaktivnih letal, z ročnimi metodami in se potem čudimo, da imamo težave. Nagel razvoj naprednih avtomatskih orodij za zasnovo in gradnjo informacijskih sistemov in sistemov za podporo odločanju spreminja tudi razvojne metodologije, ki vključujejo uvodoma navedene osnovne kriterije.

Prepad med modernimi podjetji, ki gradijo svoje informacijske sisteme s pomočjo sodobnih razvojnih metodologij, in podjetji, ki vztrajajo ali šele uvajajo na zastarel grajene aplikacije, se povečuje. Podjetja s sodobno podporo odločanju bodo ostala konkurenčna, medtem ko je usoda drugih neznana.

3. RAZVOJNE METODOLOGIJE DELA

3.1. Kaj je metodologija?

Zasnova in razvoj informacijskih sistemov sta praviloma opredeljena kot projekt, v katerem sodeluje po več tednov, mescev ali celo let interdisciplinarno usmerjen tim sodelavcev. Projektni cilji bodo doseženi, če bodo naloge in odgovornosti pravilno razdeljene in koordinirane. To lahko dosežemo na vsaj dva načina. Eden je ta, da privzamemo neformalno in intuitivno metodo ustaljenih navad in zdavega razuma (UNIZR). Pri drugem se odločimo za formalno in sistematično metologijo in sledimo njenim navodilom. Taka metodologija razdeli razvojni proces sistema na večje število dobro definiranih nalog in nudi ustrezne tehnike za njihovo realizacijo.

Metodologije si običajno pomagajo z modeliranjem dejanskih razmer tako, da je iz modela razvidna sistemska struktura. Med seboj se občutno razlikujejo, tako v stilu dela, kakor tudi zahtevanih znanjih in izkušnjah posameznih snovalcev.

Bistvena razlika med metodologijo UNIZR in neko formalno razvojno metodologijo je v tem, da imajo metodo UNIZR le nekateri eksperti v svojih glavah, medtem ko so formalne metodologije eksplicitne. Formalno metodologijo se da naučiti, UNIZR pa lahko osvojimo le s poskušanjem in odpravljanjem napak. Formalno metodologijo uporabljajo vsi člani tima, zato je zasnova sistema konsistentna, medtem ko je uporaba metode UNIZR zelo riskantna, ker je uspeh odvisen od sreče in genialnosti članov tima. Sodobne metodologije omogočajo uporabnikom neposredni vpogled v nastajanje sistema in temeljijo na uporabi avtomatskih orodij.

3.2. Metodologije strukturirane analize in zasnove

Strukturirane tehnike so se razvile iz strukturiranega programiranja, ki je nastalo v začetku 70.let kot posledica zahtev po bolj discipliniranem programiranju. Ker so ugotovili, da je še večji problem zasnova sistema oz. analiza problema, so nastale tehnike strukturirane zasnove (sredi 70.let) in strukturirane analize (konec 70.let), ki so prerasle v prave metodologije.

Te metodologije so ogromno prispevale, da je prišlo do profesionalnega, disciplinirranega pristopa pri opredelitvi problemov in iskanju optimalnih rešitev v snovanju informacijskih sistemov že konec 70. let.Vse metode so bile razvite na ročnih postopkih. Nekatere so temeljile le na analizi procesov in zato niso mogle zagotoviti izgradnje RZIS, ker so lahko zbrale le podatke za posamezne aplikacije.

V začetku 80.let je nizka produktivnost v programiranju dosegla dimenzije prave krize.Ker se je začela naglo širiti uporaba mikroračunalnikov, so mnogi uporabniki postali računalniško izobraženi in so zahtevali hitrejše odpravljanje zaostankov pri izdelavi aplikacij. V računalniških centrih (RC) so prihajali do vedno večjih vzdrževalnih problemov, četudi so obvladovali strukturirane metodologije dela. Iskanje boljših načinov za dvig produktivnosti dela je pripeljalo do pojava neprocedurnih jezikov 4. generacije, do generatorjev poročil, aplikacijskih generatorjev, do orodij za pomoč pri delu s podatkovnimi bazami (PB). Pojavili so se prvi ekspertni sistemi in sistemi za podporo odločanju. Za uporabnike so začeli izdelovati različna orodja, ki so tekla na osebnih računalnikih. Potreba po višjih nivojih avtomatizacije je postala očitna. Avtomatska orodja se pojavijo sredi 80.let kot avtomatizirane "garniture orodij" (workbench).Tiste strukturirane metode, ki so se prilagodile uporabi avtomatskih orodij, so zelo povečale produktivnost vseh uporabnikov (med takšne sodi gotovo informacijsko inženirstvo, ki ima glede gradnje poslovnih informacijskih sistemov še to prednost, da je v osnovi podatkovno orientirano).

Za vse strukturirane metodologije je značilno, da upoštevajo v svojem življenskem ciklu vsaj štiri faze: analizo, zasnovo, implementacijo (programiranje in kodiranje) in testiranje.V fazi analize je treba opredeliti problemske potrebe in izdelati predlog rešitve. Najprej študirajo problem, določijo njegove dele in medsebojne odvisnosti. Izhajajo iz zahteve, da je izredno pomembno dobro razumeti problem in šele nato predlagati rešitev. Rezultati analize morajo biti specifikacije sistema, ki opišejo, kako bo sistem zadovoljil rešitev problema.

V specifikacijah so zbrane opredelitve poročil, podatkovnih struktur, podatkovnih baz, datotek, tabel, funkcijskih komponent in povezav teh komponent. Specifikacije bi morale biti precizne, preverljive in čim bolj formalne. Pomembnost faze zasnove se često zanemarja, čeprav predstavlja podroben načrt, kako bo sistem implementiran.

3.2.1. Postopkovno orientirane metodologije

Tipična predstavnica je metodologija,ki so jo razvili DeMarco, Yourdon in Constantine.V strukturni analizi zasleduje koncept razgradnje z vrha navzdol (deli tako dolgo, da bo problem obvladan), za dokumentacijo in komunikacijo pa uporablja grafična orodja. Sistemske zahteve opredeli torej s funkcionalno dekompozicijo z vrha navzdol. Sistemske specifikacije predstavljajo z vrha navzdol razdeljen model sistema, ki naj bo zgrajen, kakor tudi povezavo analize z zasnovo. Sestavljene so iz diagramov podatkov- nih tokov, podatkovnega slovarja in procesnih določil.

Diagram podatkovnih tokov (DPT) je mrežna predstavitev sistema, tako da prikaže procese (funkcije, procedure) v sistemu in podatke, ki povezujejo te procese. Pokaže kaj sistem počne, ne pa kako. Je osrednje orodje modeliranja strukturirane analize.

Podatkovni slovar služi za povečanje rigoroznosti, ker DPT omogočajo le informativni opis sistema.V podatkovnem slovarju so

zbrani vsi podatki iz DPT ali iz izvorov/ponorov podatkov. Pri tem je treba upoštevati zbirko predpisov, po katerih so podatki urejeni.

Procesna specifikacija opisuje, kaj se zgodi v posameznem DPT. Opisan je vhod podatkov in njihova transformacija v izhodne podatke. Opis je običajno v psevdokodi (ali strukturiranem jeziku).

V strukturirani zasnovi izhajajo iz DPT,ki jih razdelajo naprej v večnivojske DPT,in iz podatkov, ki so zbrani v podatkovnem slovarju. Pri opisovanju procesov upoštevajo tudi poslovna pravila in izdelajo strukturne diagrame. Posamezni procesi so podrobno opisani v psevdokodi in s pomočjo odločitvenih tabel (dreves). Pri zasnovi je pomembna transformacijska analiza, ki začne z opredelitvijo vhoda, procesa in izhoda na podlagi DPT. Veliko popularnost je dosegla ta metoda ravno zato, ker so načela podatkovnih tokov in transformacije razumljiva in so jih zato tudi uporabniki zlahka sprejeli, kar je zlasti važno v fazi vrednotenja, ki tudi predstavlja osnovo za eventualne iteracije.

Metoda ne precizira kako konvertirati sistemsko specifikacijo v psevdokodo (programski jezik). Tudi podatkovni tokovi in transformacije niso enostavno opredeljeni (različni snovalci bodo prišli do različnih rešitev). Največja pomanjkljivost metode pa je gotovo ta, da upošteva le podatke, ki jih procesi potrebujejo. Zato ne more enolično integrirati različnih sistemov. S tem je uporabnost metodologije omejena na enostavnejše sisteme. Zaradi velike popularnosti in odlične podpore številnih konzultantskih hiš, jo skušajo prilagojevati novim zahtevam, zlasti glede snovanja direktnih in transakcijskih sistemov.

3.2.2. Podatkovno orientirane metodologije

Jacksonova metodologija, ki je zlasti razširjena v Evropi, je tipičen predstavnik podatkovno orientiranih strukturiranih metodologij. Njene premise so:

- opredelitev strukture vhodnih podatkov in izhodnih poročil
- opredelitev programske strukture, ki temelji na podatkovnih strukturah
- metode za oceno pravilnosti zasnove.

Jacksonova metodologija je dokaj formalizirana. V fazi zasnove izdela najprej diagram podatkovne strukture, ki je hierarhičen. Podatkovne strukture vgradi v programsko strukturo. Nato sledi opis operacij, ki so potrebne, da formirajo izhod na podlagi vhodnih podatkov. Programsko strukturo opiše s formalizirano psevdokodo.

Pri svojem delu uporablja tri diagramske tehnike in sicer sistemski mrežni diagram za opis podatkovnih tokov med programi, diagram drevesne strukture za predstavitev programskih in podatkovnih struktur ter strukturni tekst za opis formalne kode.

Metodologija zahteva veliko veščin in je težko obvladljiva. Žal je primerna predvsem za zasnovo sistemov, ki imajo opraviti le s sekvenčnimi datotekami. Tako nastopijo težave že, če se strukturi vhoda in izhoda ne ujemata. Podobno kot lahko nastopijo težave pri procedurno orientiranih metodah, če ne obstoji analogni (ročni) sistem, ki je izhodišče za diagrame podatkovnih tokov, lahko nastopijo težave pri Jacksonovi metodi, če niso vnaprej znane podatkovne strukture bodočega sistema.

3.2.3. Metodologija informacijskega inženirstva

Metodologija informacijskega inženirstva (I.I.) sodi gotovo med tiste metodologije, ki so se najbolj približale uvodoma postavljenim zahtevam o lastnostih sodobnih metodologij in pridejo v poštev pri gradnji RZIS. Osnovno izhodišče I.I. je, da so podatki v centru pozornosti pri moderni obdelavi podatkov. Podatke kreiramo, shranjujemo in vzdržujemo s pomočjo različnih podatkovnih sistemov. Ob zajemanju podatkov je treba opraviti primerno natančne kontrole. Ažuriranje podatkov je vsakodnevno ali periodično. Tako pripravljeni podatki služijo za kreiranje

najrazličnejših vrst informacij, kot so n.pr. rutinska poročila (dnevniki prometa, analitičnih knjigovodstev, ...), povzetki, analize in poročila za vodenje, analize "kaj-če" in informacije za pomoč pri odločanju.

Podatke lahko upravljajo različni podatkovni sistemi. Tudi shranjeni so lahko na različne načine. Lahko so porazdeljeni. Pogosto jih ažurirajo in/ali uporabljajo s pomočjo terminalov in prenosa podatkov. Drugo izhodišče I.I. je, da se podatki, ki jih uporabljajo IS, le redko spreminjajo. Entitete (to je vse tisto, o čemer zbiramo in hranimo podatke), oz. entitetni tipi se ne spreminjajo, razen v redkih primerih, ko se jim doda kak novi entitetni tip. Tudi atributi (opisujejo lastnosti entitet), oz. atributni tipi se le redko spreminjajo. Vrednosti podatkov, oz. atributov pa se stalno spreminjajo, njihova struktura pa ostaja nespremenjena, če je bila v času zasnove dobro definirana.

Čeprav so podatki relativno stabilni, pa se procedure, ki te podatke uporabljajo, pogosto spreminjajo. Zato morajo biti izdelane tako, da so maksimalno fleksibilne, ker jih sicer ni mogoče sproti prilagajati vedno novim potrebam.

Ugotavljamo torej, da se procedure poslovanja hitro spreminjajo. Podobno velja za računalniške programe, za poslovne procese, za računalniške mreže, kakor tudi za samo strojno opremo. Pri tem ostajajo le osnovni podatkovni tipi relativno stabilni.

I.I. običajno definirajo kot tesno povezano množico formalnih tehnik, s pomočjo katerih so zgrajeni podatkovno orientirani informacijski sistemi. To definicijo lahko tudi malo obrnemo: I.I. je množica avtomatiziranih metod, ki so razširjene v delovni organizaciji in jih uporabljajo za generiranje pravih informacij za prave ljudi, ob pravem času in na pravem kraju.

Med razvojne faze I.I. štejemo strateško planiranje informacijskih potreb, analizo poslovnih področij, sistemsko zasnovo in izdelavo (konstrukcijo) programov. Posebej velja omeniti fazo vzdrževanja delujočih sistemov,ki je prisotna ves čas življenja

sistemov. Od boljše ali slabše izgradnje RZIS je odvisna cena vzdrževanja, ki pri nesodobnem načinu dela angažira večino razpoložljivih kapacitet računalniških centrov. Eden od ključnih konceptov I.I.je enciklopedija-baza znanja, ki predstavlja centralno skladišče znanja o podjetju, o njegovih ciljih, strategijah,funkcijah, procesih, procedurah in programih. V njej so shranjeni podatkovni in postopkovni modeli, planske informacije, dejstva, pravila in politika, ki urejajo podjetje in njene sisteme. Kakor napreduje delo v posameznih razvojnih fazah, tako narašča znanje o podjetju.

4. NAČRTOVANJE RAZVOJA I S IN FAZE DELA

Načrtovanje IS za celotni poslovni sistem (pogosto ga imenujejo integralni informacijski sistem) je izredno pomembna aktivnost, ki jo pogosto zanemarjajo in prihajajo zato do velikih težav pri sami gradnji in uvajanju sistemov. Ugotovili smo že, da je integralni informacijski sistem zelo kompleksen, zato ga ni mogoče graditi istočasno. Posamezni podsistemi pa bodo združljivi le, če bo pred njihovo gradnjo izdelan dober in dosleden načrt razvoja integralnega IS (pravimo mu tudi arhitektura IS).

Načrtovanje IS, kakor tudi posamezne faze dela pri razvoju IS, bomo opisali s pomočjo metodologije informacijskega inženirstva, ki se vedno bolj uveljavlja. Eden od razlogov za to je gotovo v tem, da je I.I. nastalo v svoji dokončni obliki šele sredi tega desetletja in da je zamišljeno kot metodologija, ki je podprta z avtomatskimi orodji.

4.1.Strateško planiranje informacij

Uvajanje I.I. v podjetje se običajno prične z izdelavo strateškega plana informacijskih potreb. Prvi korak pri tem je

gradnja preglednega modela podjetja. Z analizo strateških ciljev in problemov dobimo strukturiran pregled ciljev in problemov, ki se jih da razgraditi in povezati z organizacijskimi enotami.Cilji morajo biti merljivi in povezani z informacijskimi potrebami. Sledi analiza tehnoloških vplivov na razvoj in konkurenčnost poslovanja. Pri tem ugotavljajo kako bo nova tehnologija vplivala na nove proizvode oz. storitve, na organizacijsko strukturo, kakor tudi na nove možnosti in nove nevarnosti, ki jih prinaša.

V nekaterih podjetjih ugotavljajo, da lahko postane poslovanje veliko uspešnejše, če se koncentrirajo na nekatera kritična področja. V analizi kritičnih faktorjev uspešnosti določijo najprej taka področja, kjer morajo stvari uspevati, nato ugotove kritične informacijske potrebe in še tiste kritične odločitve, ki potrebujejo sisteme za pomoč pri odločanju. Pregledni postop kovni model hierarhično upodablja poslovne funkcije. Povežejo jih z organizacijskimi enotami, lokacijami in entitetami. Upodobitve prikazujejo ustrezne matrike.

Na koncu je izdelan entitetno odvisnostni (na kratko entitetni) model. V entitetnem diagramu so prikazane entitete in njihove odvisnosti, kar daje pregled nad podatki, ki morajo biti hranjeni v podatkovnih bazah podjetja. V posebni matriki so entite povezane s poslovnimi funkcijami. Matriko posebej uredijo in tako najdejo naravno kohezivne grupe entitet in funkcij. Tako pridejo do pregledne informacijske arhitekture, ki je bila zasnovana neodvisno od trenutne organiziranosti podjetja. Informacija je vir podjetja, zato jo je treba načrtovati za vse podjetje. Implementacija arhitekture pa bo odrazila trenutno organiziranost, kar je pomembno pri določanju priòritet za nadaljnje delo.

4.2. Analiza poslovnih področij

Analiza poslovnega področja oskrbi jasno razumevanje poslovanja na izbranem področju in formira arhitekturni okvir za gradnjo RZIS. Tako postavljen okvir zagotavlja, da so neodvisno razviti

sistemi povsem prilagojeni. Okvir sestavljajo:

i) podatkovni model, ki je osnovni gradnik zasnove in izdelave aplikacij
ii) model poslovnih postopkov in njihovih medsebojnih odvisnosti
iii) povezava obeh modelov, ki pokaže kateri postopki uporabljajo katere podatke.

Podatkovni model predstavlja logično strukturo podatkovnih elementov in njihovih medsebojnih odvisnosti tako, da je povsem neodvisna od računalnikov, programske opreme in od procedur, ki uporabljajo podatke, ker se vsi ti parametri pogosto spreminjajo. Podatkovno modeliranje predstavlja postopek sinteze različnih podatkov. Tako zberejo snovalci podatke iz različnih aplikacij, iz dokumentov, kakor tudi informacijske potrebe poslovodstva in jih sintetizirajo. Tehnika sinteze temelji na naboru formalnih pravil tako, da je na koncu izdelan popolnoma normaliziran podatkovni model. Proces normalizacije je definiran kot množica matematičnih pravil, ki odpravijo odvečnost v podatkovnih strukturah, zagotovijo odvisnost vsakega atributa v zapisu od ključa (od celotnega ključa in od ničesar drugega) in minimizirajo vzdrževalne stroške podatkov. baz.Uporabniki in snovalci preverijo model in odpravijo protislovja. Sledi stabilnostna analiza, ki zagotavlja stabilni model. Pri malo obsežnejših področjih je treba vpeljati avtomatska orodja, ker postane ročno delo preveč zamudno in nepregledno.

Pri izdelavi postopkovnega modela najprej identificirajo poslovne funkcije in jih s pomočjo dekompozicijskega diagrama razstavijo na enostavnejše procese. Medsebojne odvisnosti med procesi raziščejo s pomočjo diagramov odvisnosti oz. diagramov podatkovnih tokov. Z matriko odvisnosti procesov in podatkov ponazorijo povezavo postopkov in podatkov. Iz nje razberemo kateri postopki kreirajo, ažurirajo, brišejo ali berejo določene podatke.

Povezavo s strateškim planiranjem oskrbi enciklopedija.Tudi vso informacijo o podatkovnem in postopkovnem modelu, ki je predstavljena s pomočjo različnih diagramov, shranimo in ažuriramo v

enciklopediji.Tako je po koncu te faze dela v enciklopediji zbranega že veliko znanja o podjetju, njegovih ciljih, podatkih in postopkih. Analiza poslovnega področja ima naslednje značilnosti:

- je izvajana za vsako poslovno področje posebej
- potrebuje intenzivno sodelovanje uporabnikov
- je neodvisna od informacijske tehnologije
- je neodvisna od tekočih sistemov in postopkov
- često zahteva premislek in ponovno postavitev sistemov in postopkov
- določa področja za sistemsko zasnovo
- kreira podrobni podatkovni model za dano področje
- kreira podrobni postopkovni model in ga poveže s podatkovnim modelom (pri tem upošteva vpliv bodočnosti in okolja)
- rezultati so zbrani in vzdrževani v enciklopediji.

4.3. Zasnova in izdelava sistemov

V zadnjih dveh fazah se snovalci lotijo zasnove in izdelave aplikacijskih sistemov. Kot osnova jim služi izhod iz prejšnje faze, to je, podrobni podatkovni in postopkovni model z medsebojnimi povezavami.

Sistemi, ki jih je treba izdelati, se glede velikosti in kompeksnosti močno razlikujejo med seboj. Zato jih običajno razdelijo v tri kategorije glede zahtevnosti:

- sistemi, ki jih lahko izdelajo sami uporabniki ob podpori informacijskih centrov
- manjši projekti, ki jih lahko dokončajo majhni timi (običajna sestava ima enega do dva računalnikarja in enega ali dva uporabnika)
- zahtevni projekti, ki vključujejo večje time z ustrezno projektno organizacijo.

Snovalci si morajo zlasti prizadevati, da čim aktivneje vključijo uporabnike. Predpogoj za to je ustrezno šolanje, kjer se uporabniki seznanijo z vsemi uporabljanimi diagramskimi in drugimi tehnikami. Med najkreativnejše oblike sodelovanja sodi

gotovo izdelava prototipov. Tam, kjer že delujejo infomacijski centri, lahko le-ti odigrajo pomembno vlogo pri vključevanju uporabnikov, zlasti pri projektih, ki jih lahko zasnujejo in izdelajo sami uporabniki. Pri zasnovi zahtevnejših projektov uporabljajo v zadnjem času metodo skupne zasnove aplikacij (SZA). S to metodo izdelujejo sisteme, ki v celoti izpolnjujejo uporabniške zahteve (metodo so prvič vpeljali pri IBM). Kadar metodo povežejo z informacijskim inženirstvom in z metodo izdelave prototipov, postane izredno učinkovita. Še več, vzpodbuja kreativnost uporabnikov, zato so rešitve inovativne.

Naloge snovalcev v fazi zasnove na podatkovni strani so:

- ekstrakcija ustreznega podatkovnega podmodela iz globalnega podatkovnega modela
- izdelava podatkovnega strukturnega diagrama
- pri zahtevnih projektih, ki so interaktivne narave, sledi analiza rabe poti, porazdelitvena analiza in šele nato fizična zasnova podatkovne baze.

Na postopkovni strani je treba zasnovati in izdelati procedure aplikacijskega sistema.Procedure razvijajo na podatkovnem podmodelu. Pri tem uporabljajo različne vrste diagramov, ki jih tesno povezuje enciklopedija. Uporabni so predvsem podatkovni navigacijski diagrami (omogočajo navigacijo in ustrezno akcijo nad podatki, dopuščajo pa tudi določanje sekvence in izbora pogojev), akcijski diagrami (to so splošno uporabni diagrami, tako za pregledne kot podrobne opise) in odločitvena drevesa ali odločitvene tabele. Četudi uporabljajo druge diagramske tehnike, morajo imeti snovalci pred očmi zahtevo, da si izberejo le take vrste diagamov, ki se jih da avtomatsko prevesti v akcijske diagrame. Z akcijskimi diagrami se namreč najbolje opišejo podrobna programska določila, ki so osnova tako za izdelavo prototipov kot tudi za vključitev kakšnega od jezikov 4. generacije oz. 3. generacije, če to narekujejo okoliščine.

V tej fazi načrtajo tudi izglede ekranov, poročil, zasnujejo pa tudi dialoge človek-stroj. Snovalci lahko svoje delo izredno pospešijo, če uporabljajo pri delu avtomatska orodja. Avtomatska

grafična orodja pomagajo pri risanju,organizaciji in spreminjanju diagramov.Zahtevajo eksaktnost in omogočijo avtomatsko konverzijo različnih diagramov v akcijske diagrame.Tako ostane zasnova neodvisna od specifičnega programskega jezika, dokler je mogoče.

V fazi izdelave in implementacije programov je treba oskrbeti različne poglede na podatke in skodirati akcijske diagrame ali prototipe. Temu sledi podrobno testiranje. Če je na voljo generator kode, izdelajo kodo z njegovo pomočjo.

4.4. Pregled nekaterih značilnosti in prednosti I.I.

Dosedanje izkušnje z I.I. kažejo na to, da je mogoče sisteme izredno hitro izdelati, ko so bili zgrajeni podrobni podatkovni modeli. To pomeni, da zahteva I.I. veliko večjo angažiranost v fazah planiranja, analize in zasnove kot je to primer s konvencionalnimi metodami.Z razvojem in uvajanjem vedno boljših avtomatskih orodij še pospešijo izdelavo sistemov. V tistih centrih, kjer jim je uspelo povezati avtomatsko zasnovo z generatorji kode so drastično povečali produktivnost dela.

Metodologija I.I. je povezana s strateškim planiranjem poslovnega sistema. Zato in zaradi angažiranosti celotne organizacije na realizaciji projekta, zahteva popolno podporo in usmerjanje poslovodstva, sicer ni mogoče zagotoviti popolnega uspeha. Kljub temu se vedno več organizacij v razvitem svetu odloča za uvedbo te metodologije, ker so njene prednosti pred drugimi velike.

Naštejmo nekatere pomembnejše prednosti I.I. Metodologija podpira informacijske potrebe poslovodstva in samouravnih organov in usmerja obdelavo podatkov na poslovne cilje. Vključuje uporabnike v planiranje, analizo in zasnovo sistemov. Tako zgrajene sisteme zlahka modificirajo, s tem pa odpravijo velike probleme vzdrževanja.Sistemi so integrirani v okviru celotne organizacije, ker je pristop rigorozen. Metodologija I.I. omogoči veliko boljše razumevanje in centralno kontrolo nad podatki kot enim od najpomembnejših virov podjetja.

Tolikšne prednosti so možne le zaradi celovitega pristopa metodologije h gradnji informacijskih sistemov. Celovit pristop ji omogočajo njene značilnosti:

- je uporabniško orientirana
- temelji na lahko razumljivih diagramih
- teži k popolni avtomatizaciji in povezuje avtomatizacijo zasnove z generatorji kode in jeziki 4. generacije povsod, kjer je to smiselno
- povečuje hitrost razvoja obdelav in vrednost rač. sistemov
- uporablja pristop prototipov
- pomaga informacijskemu centru pri delu z uporabniki
- je integrirana v vseh svojih razvojnih fazah dela.

5. UPOSTEVANJE UPORABNIŠKIH ZAHTEV

5.1. Uporabniško orientirane razvojne tehnike

5.1.1. Problemi tradicionalnih metod

Tradicionalne metode so bile uspešne pri razvoju aplikacij, ki so bile orientirane na papirno obdelavo (n.pr. obračun plač, saldakontov). Te metode niso bile uspešne pri gradnji sistemov za podporo odločanju. Bile so primerne za paketne obdelave, ne pa za gradnjo interaktivnih sistemov.
Kako bolje zadostiti uporabniškim zahtevam v novih sistemih? Tako, da s pomočjo novih metod vključimo kreativne uporabnike že v proces zasnove. Tradicionalne tehnike so zlasti pomanjkljive v fazah analize in zasnove, kjer nastane največ napak. Rašitev tega problema predstavljajo sodobne tehnike kot so prototipni pristop, skupna zasnova aplikacij, metodologija časovnih omejitev in informacijski centri (uporabnikov in poslovodstva).

5.1.2. Prototipni pristop

Tehnika gradnje prototipov je primerna za izdelavo grobe verzije želenega sistema ali njegovih delov.Prototip opisuje sistem in služi za pregled predlagane zasnove, saj je primernejši kot papirne specifikacije. Predvsem je nekaj realnega in uporabniki si lahko predstavljajo kakšen bo njihov dokončni sistem.

Tehniki izdelave prototipov se posveča vedno več pozornosti in postaja glavna metoda pri uporabniško usmerjenem razvoju. Ločiti pa moramo med industrijsko izdelavo prototipov in izdelavo prototipa za programsko opremo.Industrijski prototip je dražji kot končni proizvod in njegova izdelava je dolgotrajnejša. Izdelava prototipa programske opreme je smiselna, če je poceni in kratkotrajna. Zato pa praviloma ni končni proizvod, ampak predstavlja poenostavljeno verzijo sistema, ki sicer vsebuje večino funkcij, ne upošteva pa učinkovitosti, pa tudi ne lastnosti kot so varnost, obnovljivost, zmožnost ravnanja z veliko količino podatkov ali z veliko uporabniki.

5.1.3. Skupna zasnova aplikacij

Tehniko skupna zasnova aplikacij (SZA) so razvili pri IBM Kanada. Povsod, kjer so jo uporabili, so pospešili proces analize potreb in zasnovo sistema. Osnovna ideja SZA je v tem, da se izbrani ključni uporabniki in profesionalci za informacijski sistem (IS) pod posebnim vodstvom zberejo na izločenih sestankih, kjer po predvidenih korakih izdelajo sistemsko zasnovo in morda tudi prototip (gotovo pa delne prototipe). Metoda je postala zlasti uspešna z uporabo avtomatskih orodij v okviru I.I.

Seje SZA naj trajajo od nekaj dni do enega tedna in to izven sedeža organizacije, da ni nepotrebnih motenj. Kadar gre za kompleksne sisteme, lahko seje trajajo dalj časa. V takih primerih je priporočljivo razbiti aplikacijski sistem na podsisteme, če je na razpolago ustrezno orodje, ki zagotavlja

konsistentnost podatkovnih modelov in natančne vmesnike med podsistemi.

Vodenje sej SZA je zahtevno opravilo. Od vodje pričakujemo, da je ustrezno šolan, zlasti na področjih komuniciranja in pogajanj, na področju analize in zasnove, na področju podatkovnega modeliranja in pri poznavanju ustreznih diagramskih tehnik, pri uporabi avtomatskih orodij in seveda na področju vodenja projektov. Kadar niso prisotni ključni uporabniki, naj seje odpadejo in se o tem obvesti vodstvo podjetja. Tehnika SZA je zlasti primerna, če gre za aplikacije z več lokacijami ali za večdisciplinarna področja.

5.1.4. Metodologija časovnih omejitev

Velike spremembe v zasnovi in uvajanju sistemov so povzročile ravno tehnike kot je SZA, izdelava prototipov, avtomatska orodja zasnove in generatorji kod. V okviru I.I. je bila izdelana metodologija časovnih omejitev (okvirov), ki ustrezno povezuje vse štiri naštete tehnike. V danem časovnem okviru mora biti zgrajen delujoči sistem. Tako se izognemo nevarnosti, ki je pogosto prisotna pri izdelavi prototipov ali drugih iterativnih tehnikah, da funkcije sistema nekontrolirano naraščajo.

Predviden časovni okvir ni raztegljiv. Sistem mora biti v danem časovnem okviru zgrajen do te mere, da ga je mogoče vpeljati. V tem okviru poteka sicer kontinuiran iterativni razvoj, vendar so razvijalci v časovni stiski, zato pazijo tudi na dodajanje novih funkcij sistema. Metodologijo časovnih omejitev je smiselno vpeljati tam, kjer je na voljo močno, učinkovito in enostavno razvojno orodje. Orodje mora zagotavljati enostavno izdelavo prototipov, tako da prototip lahko preraste v delujoči sistem. Orodje mora biti tudi enostavno za uporabo. da lahko uporabniki sodelujejo v procesu zasnove.

Tipična dolžina časovnega okvira je 60 do 90 dni. Delovni tim ne sme biti velik. Primerna velikost je od 2 do 6 oseb (pri kompleksnem razvoju je tim lahko tudi večji). Splošni koncept metodologije časovnih omejitev je, da je bolje imeti delujoči

sistem omejene funkcionalnosti v kratkem času kot pa čakati na vsestranski sistem nekaj let. Tudi na omejenem sistemu si lahko naberemo veliko izkušenj, nato pa ga razširimo z novim časovnim okvirom. Če se izkaže, da je bil sistem napačno zamišljen, je bolje to spoznati v 2 - 3 mescih kot ugotavljati kaj podobnega po dolgem času (to se pogosto dogaja pri tradicionalnem razvoju).

5.1.5. Informacijski centri

Informacijske centre (IC) ustanavljajo povsod tam, kjer želijo vzpodbuditi, izšolati in podpreti uporabnike, ki uporabljajo računalnike neposredno, bodisi za izdelavo poročil ali za izdelavo lastnih aplikacij. Glavne naloge IC bi morale biti: vzpodbujati boljše odločanje s pomočjo uporabe računalniških orodij in informacij, izboljšati produktivnost dela šolanih delavcev, obiti zaostanke pri razvoju sistemov v računalniških centrih, vzpodbujati uporabo boljših postopkov.

V splošnem naj uporabniki grade sisteme, ki niso zahtevni, ki jih uporabljajo le v eni organizacijski enoti in ki rešujejo njihove lastne probleme. Dostop do produkcijskih podatkovnih baz (PB) naj bo koordiniran.Posebni programi - ekstraktorji pripravljajo za uporabnike različne PB, posebej za poizvedovanje, za poročanje in posebne PB za sisteme za podporo odločanju. Namen IC je hitro priti do rezultatov. Če ni takojšnjih rezultatov, lahko ideja o IC propade. Zato je važno, da so prvi uporabniki dobro izbrani. Biti morajo taki, ki potrebujejo takojšnjo rešitev svojega poslovnega problema in so zato pripravljeni tudi na določene napore.

5.2. Problemi razvojnega življenjskega cikla

Zgodovinski razvojni življenjski cikel obsega praviloma naslednje faze dela: sistemska analiza, sistemske specifikacije, zasnova, programiranje, testiranje, integracija in vpeljava. Tak cikel temelji na metodologijah, ki so nastale pred 1985. Zanje je

značilno, da so bile ročne. Posamezne faze dela so se zaključile same vase, tako da je celotni cikel spominjal na kaskadni slap. Zahteval je veliko ročnega dela, zato je zlahka prišlo do napak. Vsaka faza je zahtevala svoj jezik. Pri opredelitvi sistemskih specifikacij je bilo premalo formalnosti. Uporabljena orodja so bila zahtevna, vendar nekonsistentna. Verifikacija in vrednotenje sistema je bilo možno šele na koncu cikla (v fazi testiranja).

Po l. 1985 so nastale metodologije, ki uvajajo avtomatska razvojna orodja, se izogibajo ročnemu kodiranju, kjer je to smiselno, vpeljujejo avtomatsko zasnovo in programiranje. Pravilnost zasnove preverjajo računalniško.Današnja gradnja kvalitetnih sistemov zahteva tolikšno natančnost, da je ročne metode ne morejo ponuditi. Zato je razumljiva zahteva po izboru kvalitetne in sodobne metodologije, ki je podprta z avtomatskimi razvojnimi orodji.

Pojavilo se je več vrst življenjskih ciklov, ki bolje zadostijo potrebam kot klasični cikel:

- Življenjski cikel z enojno iteracijo. Sem sodi tudi klasični cikel, vendar danes uporabljajo orodja za izdelavo prototipov, za avtomatsko zasnovo, generatorje kod, kar je spremenilo njegovo naravo.
- Evolucijski življenjski cikel. V tem ciklu nastaja zaporedje prototipov, ki se bližajo ciljnemu sistemu. Končni prototip lahko preraste v delujoči sistem.
- Življenjski cikel časovnih omejitev. To je zelo uspešni način kako vključiti izdelavo prototipov, tehniko SZA, avtomatsko zasnovo in generacijo kode. Tako je pospešen razvoj, stroški in riziki razvoja pa so minimizirani.
- Življenjski cikel hitrih rezultatov. Uporabljan je pri razvoju enostavnih aplikacij. Ne potrebuje niti formalnih specifikacij niti zasnove. Uporabljajo ga zlasti v IC.
- Življenjski cikel za razvoj sistemov za podporo odločanju. Sistem za podporo odločanju pomaga pri boljših poslovnih odločitvah na določenem področju. Razvijajo jih praviloma uporabniki na inovativen način, brez specifikacij.

6. SISTEMI ZA IZBOLJŠANO ODLOČANJE

6.1. Vrste sistemov

Posebno koristna uporaba računalnikov omogoča izboljšano odločanje v podjetju. Običajno delijo sisteme, ki omogočajo boljše odločanje, na eksekutivne informacijske sisteme (EIS) in na sisteme za podporo odločanju (DSS - pride od angleškega izraza "Decision Support Systems").

EIS so namenjeni poslovodstvu, ki ne utegne izdelovati podrobnih analiz o odločanju. Hoče vedeti kaj se dogaja, določa kdaj je potrebna človeško posredovanje, potrebuje pravo informacijo, da se lahko pravilno odloči in krmili učinke svojih odločitev. EIS morajo biti enostavni za uporabo in neposredno koristni poslovodstvu.

Sistemi za podporo odločanju (DSS) so namenjeni tistim vodilnim delavcem in analitikom, ki opravljajo poglobljene analize, da bi prišli do boljših odločitev. Računalniške tehnike segajo od metod zdravega razuma do zelo sofisticiranih metod operacijskih raziskav.

6.2. Sistemi za podporo odločanju (DSS)

DSS nudijo pomoč pri iskanju podatkov za odločanje, delajo analize in modeliranja odločitev in priporočajo optimalni izbor odločitev. Paketi za DSS pomagajo uporabnikom pri analizi podatkov, pri odgovorih na vprašanja "kaj-če", kreirajo finančne in druge modele. Segajo od enostavnih, dvodimenzionalnih preglednic (n.pr. LOTUS 1-2-3) do kompleksnih, večdimenzionalnih orodij, ki omogočajo izčrpne analize.

DSS naj praviloma grade uporabniki, ki imajo opraviti z odločanjem, ker bolje razumejo probleme kot profesionalci IS. Tem

je namenjena pomoč pri izboru najustreznejšega paketa DSS, pri izbiranju potrebnih podatkov in pri gradnji podatkovnih baz. Za gradnjo dovršenih DSS je potrebno skrbno planiranje in profesionalna zasnova, medtem ko enostavni DSS premorejo malo več kot enostavna računanja na kalkulatorju. Vsa orodja DSS postanejo bolj razumljiva z uporabo računalniške grafike.

6.3. Eksekutivni (izvršni) informacijski sistemi (EIS)

Gradnja EIS je veliko enostavnejša kot gradnja sistema za podporo odločanju. Izbrati je treba tak programski paket, da ni potrebno programiranje. Vrednost EIS je velika tedaj, če jih resnično uporabljajo poslovodniki. EIS mora omogočiti poslovodniku, da hitreje odkrije potencialne težave. Sistem torej omogoča, da poslovodni delavci sodelujejo pri ključnih odločitvah, ko je to potrebno.

Ker je EIS relativno enostavno zgraditi, jih morajo popularizirati tudi vodilni delavci, ki so odgovorni za informacijski sistem. Poslovodstvo ima šele z uvedbo EIS neposredno korist od profesionalcev IS.

Kakšna je razlika med DSS in EIS?
Sistemi za podporo odločanju so zgrajeni za obsežne analize in za izdelavo modelov. Praviloma so programirljivi in imajo ad hoc pristop do podatkov. Zasnovani so tako, da omogočajo fleksibilno uporabo raznih analitičnih orodij.

Po drugi strani pa so EIS zasnovani za prikazovanje informacij in za krmiljenje situacij. Niso programirljivi in imajo vnaprej zasnovan pristop do podatkov. Zasnovani so tako, da je njihova uporaba res enostavna.

V EIS so shranjeni interni (n.pr. standardna finančna poročila, analiza prodaje, modeli in napovedi, kadri) in zunanji podatki (n.pr. analiza konkurence, analiza proizvodov, ekonomska analiza, finančne novice, industrijske vesti).

INFO-TEHNOLOGIJA PROTOTIPNEGA PRISTOPA IZGRADNJE RAČUNALNIŠKIH APLIKACIJ

D. Krstič
Iskra Zorin - Tozd CAOP

1. Uvod

V preteklih četrt stoletja je bila večina informacijskih podsistemov (računalniških obdelav) razvita v eni izmed različic t.i. življenjskega ciklusa razvoja sistema. Tipičen življenjski ciklus razvoja je sestavljen iz naslednjih stopenj, ki si v glavnem sledijo zaporedno (sekvencionalno):

1. strateško planiranje informacijskega sistema,
2. informacijska analiza potreb,
3. modeliranje podatkov (in vzporedno 4.),
4. oblikovanje postopkov oz. analiza funkcij,
5. programiranje,
6. testiranje programov in obdelave,
7. uvajanje obdelave in
8. vzdrževanje obdelave.

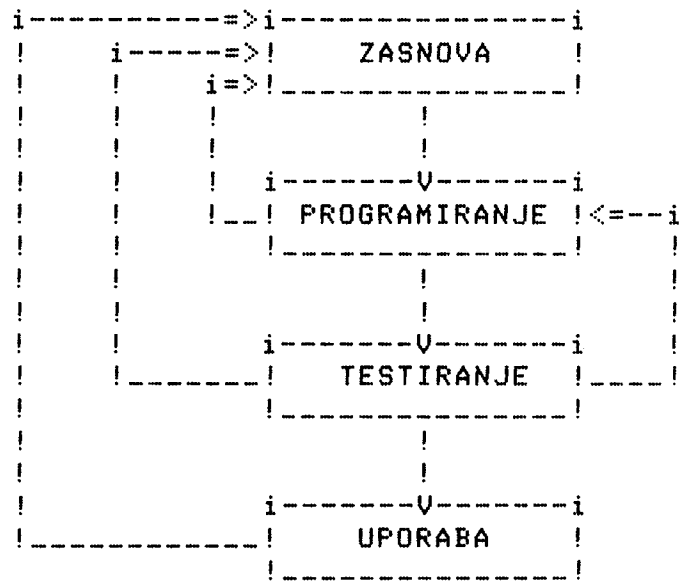
Računalniško podprti postopek, funkcijo za določeno potrebo uporabnika opredelimo z

- vhodnimi podatki,
- algoritmom,
- izhodnimi podatki ter
- dogodki,
  - ki funkcijo sprožijo oz.,
  - ki so posledica funkcije.

Ko se lotimo dela spoznavanja funkcij, moramo poznati kriterije, po katerih lahko za vsako funkcijo opredelimo, ali je relevantna za sistem informacijskih potreb uporabnika. Do podatkov o funkcijah lahko pridemo z intervjuji uporabnikov, z opazovanjem in analizo njihovega delovanja, s pregledom obstoječih poročil, planov, poslovne politike in organizacijskih predpisov. Vsekakor moramo pred nadaljevanjem dela in še zlasti programiranja, podrobno opredeliti vse funkcije in dobiti potrditev uporabnika (običajno pismeno), da se strinja s specifikacijami.

V zgoraj opisanem postopku je inherentno skrita vrsta določenih težav, ki onemogočajo natančno spoznavanje funkcij. Opazovanje delovanja uporabnika običajno ni enostavno; dostikrat tudi uporabnik ne zna podrobno opisati vse funkcije, ki jih potrebuje za svoje delo. Sistemski analitik pa spet po svoje razume poslovno okolje, ki ga analizira, nakar predoči v podpis uporabniku zasnovo bodoče obdelave, izdelane s pripomočki, kot jih uporablja sam. Tudi če je dobil podpis o strinjanju, to dejansko še ne pomeni, da je uporabnik v celoti dojel zasnovo na enak način kot sistemski analitik. Zato že v kali obstojajo določeni pomisleki, da uporabnik ne bo zadovoljen z računalniško obdelavo, ko bo narejena. To toliko bolj, ker je iz nadaljnjega ustvarjanja obdelave uporabnik skoraj v celoti izključen.

Zaradi omenjenih težav in nepopolne zasnove računalniške obdelave v življenjskem ciklusu razvoja sistema, imamo v praksi pogosto opravka z "jalovimi" povratnimi zankami, ki so neučinkovite in drage (gl. sl.1.).



Sl.1. "Jalove" povratne zanke zaradi neustrezne zasnove IS

Z gradnjo EIS je treba pričeti čim enostavneje. Koncentrirati se je treba le na nekaj vrst informacij, ki pa so največ vredne za tistega poslovodnika, ki mu gradimo EIS. Že prva delovna verzija mora biti uporabna. Tedaj lahko pričakujemo bolj podrobne informacije in/ali drugo vrsto informacij.

Sistem bo v začetku namenjen le enemu izvršnemu delavcu, postopoma pa bo razširjen tako, da bo na voljo celotnemu poslovodstvu. Pri tem je treba paziti, da EIS ne bo vseboval preveč informacij, ki jih je drago vzdrževati. Ostati mora enostaven glede uporabe in osredotočen na bistvene informacije.

Gradnja EIS ne zahteva veliko tehničnih znanj. Če je formirana posebna ekipa za pomoč pri gradnji in pozneje pri uporabi EIS, naj bo sestavljena iz strokovnjakov, ki razumejo informacijske potrebe poslovodstva in ki znajo informacije iz EIS prikazati na atraktiven način. Ta ekipa si lahko posebna računalniška znanja vedno "sposodi" v računalniškem centru.

Po B.H.Boaru (BOAR,1984) so trije izraziti in izčrpni problemi, ki jih sistemsko analitske tehnike za razvoj računalniških aplikacij niso mogle do sedaj učinkovito razrešiti:

1. Uporabniki izredno težko specificirajo in opišejo vse karakteristike končne računalniške obdelave.

2. Grafične in opisne dokumentacije zasnove niso kos dinamičnim spremembam uporabnikovih potreb in dokončni opredelitvi obdelave.

3. Nesporazumi med uporabniki in oblikovalci obdelave so endemične narave.

Za premagovanje vseh opisanih težav je bila razvita vrsta tehnik za boljšo definicijo sistema kot sta npr. metode strukturne analize (recimo metoda SADT, kar je akronim za Stuctured Analysis and Design Techniques) in specifikacijski jeziki (recimo PSL/PSA, kar je akronim za Problem Statement Language/Problem Statement Analyzer). Praksa je pokazala, da tudi te tehnike niso razrešile bistvo problema - globoke in pogoste nesporazume med oblikovalci in uporabniki obdelav.

Sredi sedamdesetih let so zgoraj omenjene tehnike sicer omogočile delen dvig produktivnosti sistemskih analitikov na osnovi strukturiranih pristopov. Toda davek za uvedbo stroge dokumentacije pri razčlenevanju funkcij je bilo izredno povečanje birokracije snovanja IS z obilico obrazcev, ki so bili uporabnikom še bolj nerazumljivi kot besedni in/ali grafični opisi sistema (CHORAFAS,1986;I).

V Iskra Zorin, Tozd CAOP smo SADT tehniko proučili konec 1.1980, toda nismo jo priporočili sistemskim analitikom za redno uporabo prav zaradi zgoraj omenjenih slabih lastnosti. V Iskra Commerce področje SOR uporablja metodo SADT od 1. 1981 dalje, v povezavi z drugimi metodami. Leta 1985 smo v CAOP (KRSTIC,1985;i) preiskusili metodo PSL/PSA v okviru projekta Zasnove informacijskega sistema v Iskri. Nismo našli nobene večje prednosti uporabe programskega paketa PSL/PSA niti za uporabnike niti za oblikovalce računalniških obdelav (sistemske analitike oz. projektante).

(BOAR,1984) Boar,B.H.: "Application Prototyping - A Requirements Definition Strategy For The 80s"; J.Wiley and Sons, New York, 1984, ss. 210, s. IX.

(CHORAFAS,1986;I) Chorafas,D.N.: "Fourth and Fifth Generation Programming Languages, Vol.1: Integrated Software, Database Languages, and Expert Systems"; McGraw - Hill Book Comp., New York, 1986, ss.283, s.66.

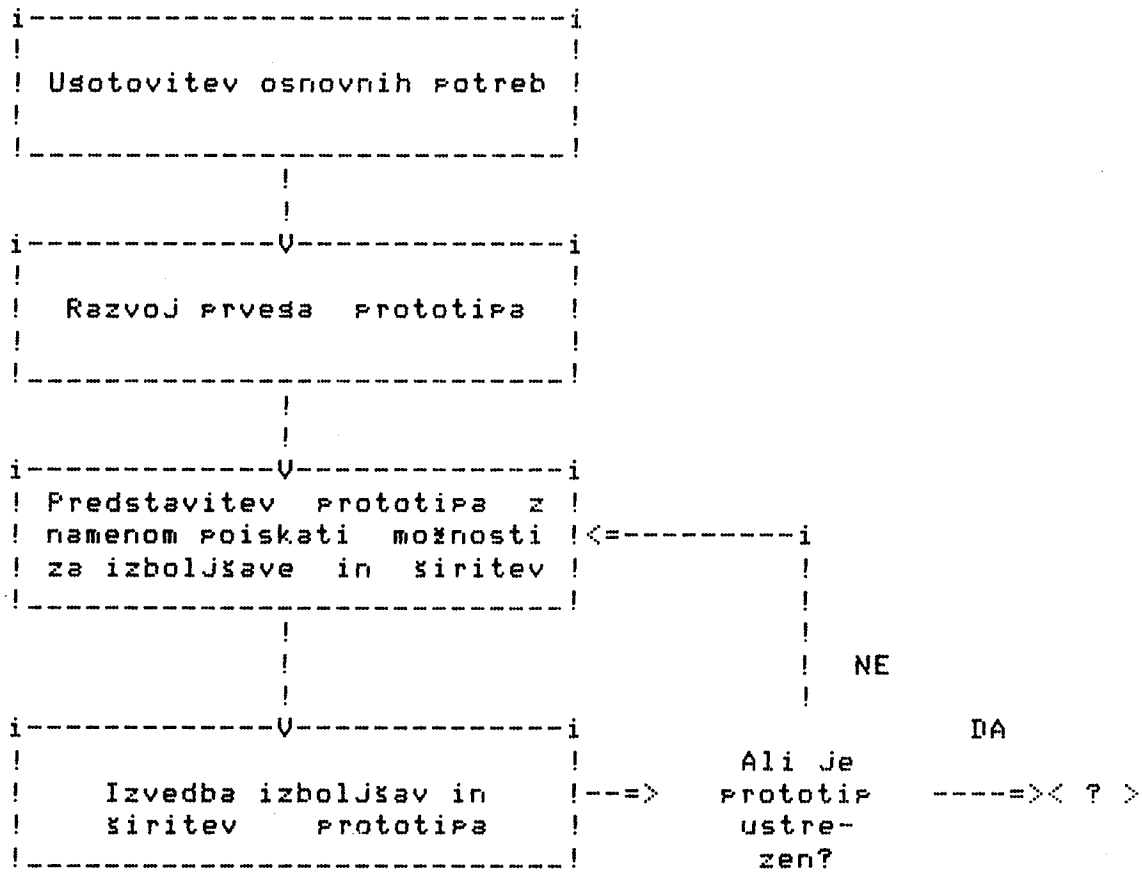
(KRSTIC,1985;i) Krstič,D.: interna gradiva v zvezi s projektom Zasnove informacijskega sistema v Iskri: ZISI-4, Tozd CAOP, Ljubljana, 1985.

Nove možnosti za premagovanje ovir, ki so značilne za življenjski ciklus razvoja sistema, je ponudil prototipni pristop izgradnje računalniških obdelav. V drugih področjih, in še zlasti v materialni proizvodnji, je prototipni pristop pri nastajanju končnega izdelka že dolgo običajna praksa. Pri izdelavi uporabniške programske opreme pa se ta metodologija uveljavlja šele zadnja leta. Razlog za tako pozno vključevanje prototipnega pristopa za razvoj programske opreme je treba iskati predvsem v nezadostni razvitosti informacijske tehnologije za tovrstne pristope še v bližnji preteklosti. Še več: tudi danes je prototipni pristop pionirski pristop, ki se je kljub nekaterim svojim prednostim šele začel uveljavljati.

2. Tehnika prototipnega pristopa

Izdelava računalniškega prototipa (angl. prototyping) je metoda za oblikovanje, predstavitev in izpopolnitev uporabnikovih potreb z izgradnjo delujočega modela končnega sistema - in sicer hitro in v tesnem sodelovanju z bodočim uporabnikom. S postopnimi iterativnimi izboljšavami (gl. sl.2.) izhodiščnega prototipa kot rezultat medsebojnega, čedalje boljšega razumevanja problema, pridemo lahko končno do prototipa, ki v bistvenih potezah zadovoljuje potrebe naročnika. Izdelava prototipov se zelo razlikuje od pristopov izgradnje računalniško podprtih informacijskih sistemov po načelu življenjskega ciklusa razvoja. Filozofija uporabnosti prototipa sloni na človeški lastnosti, da je običajno lažje kritizirati obstoječe kot ustvarjati. Potrebno je poudariti, da je prototip v bistvu poenostavljena interaktivna obdelava, običajno sprogramirana z enim izmed t.i. programskih jezikov četrte generacije (J4G) in da je razvoj prototipa v bistvu heuristično razvoj.

Možnost uporabe prototipnega pristopa v veliki meri sloni na dejstvu, da je večina poslovnih obdelav medsebojno precej podobnih po svoji naravi. Potrebno je oblikovati podatkovne baze, obstoječim podatkom dodajati nove podatke, spreminjati vsebino le-teh, jih presledovati in jih končno brisati, če jih ne rabimo. Veliko je podobnosti tudi med nekaterimi sistemskimi modeli, modeli podatkov, modeli funkcij, v zasnovi obdelav in oblikami poročil. Zato se v svetu čedalje bolj uveljavlja pristop "ponovne uporabe uporabniških programov" (angl. reusable software). Dobre programske rešitve je pač treba spraviti v knjižnico programov in jih po potrebi uporabiti v podobni situaciji. Dostikrat so potrebne le neznatne spremembe programov, ki se lahko hitro izvedejo s sodobnimi interaktivnimi urejevalniki (editorji).



Sl.2. Postopek izdelave prototipa

Mnenja smo, da je za uspešen razvoj prototipa že takoj na začetku zelo pomembno:

- čimbolj natančno ugotoviti osnovne informacijske potrebe uporabnika,
- raziskati kje so spravljeni potrebni podatki in v kakšni obliki,
- opredeliti manjkajoče podatke, jih uskladiti s podatki v bazi ter izvesti njihovo normalizacijo in
- skicirati scenarij prvega prototipa.

Zgornji odstavki v glavnem ne potrebujejo dodatne razlage. Morebiti je izjema le pojem scenarija, izraz, ki se je sicer že uveljavil v kontekstu izdelave prototipov. Mark E. Lipp

(LIPP,1985) opredeljuje scenarij kot logično zasnovo, ki ji sledimo, da dosežemo obvladovanje načrtovanih informacij. Cilj je, da bi scenarij ustrezal sledanju končnega uporabnika na potrebno ravnanje s podatki na nivoju atributov.

Nepravilno koncipirana izhodišča pred izdelavo prototipa imajo lahko pozneje daljnosežne posledice in lahko motijo vključitev prototipa oz. obdelave na njegovi osnovi v celoto informacijskega sistema. Na nevarnost zanemarjanja logičnega modeliranja potreb in rešitev v prototipnem pristopu opozarjata tudi J.Grad in A.M.Jenkins (GRAD,JENKINS,1986), kajti izdelava prototipa postavlja v ospredje predvsem fizične aspekte sistema; uporabnik pač potrebuje hitro rešitev za svoj problem in običajno nima dobrega pregleda nad celovitim informacijskim sistemom.

Izrednega pomena je, da že pri razvoju prve različice prototipa (gl. sl.2.) upoštevamo čimveč bistvenih potreb bodočega uporabnika. B.H.Boar meni (BOAR,1984), da bo uporabnik izgubil voljo delati naprej na prototipu, če je ustreznost prve različice prototipa manjša od 60%. To samo potrjuje ugotovitve iz prejšnjega odstavka o pomembnosti začetne (preliminarne) analize. Pravilno je tudi, da prva predstavitev prototipa zajame (površno) celotno vsebino uporabnikovih potreb, ne pa posamene dele podrobno in druge izpusti.

Oblikovalec prototipa se mora vsekakor potruditi, da v čimkrajšem času in v čimmanj iteracijah doseže tisto kakovost in funkcionalnost prototipa, ki bo zadovoljila uporabnika. Zato je ugodno, če ima oblikovalec prototipa priložnost, da predstavi uporabniku določene podobne (sorodne) že izdelane računalniške rešitve in s tem eventualno omogoči uporabniku, da se približno opredeli glede svojih potreb in načina dela. To lahko zmanjša tveganje, da bo prva različica prototipa popolnoma neustrezna (kar se tudi občasno dogaja).

Tipičen vrstni red pri izdelavi prototipa je lahko:

- programiranje scenarija in menujev,
- oblikovanje osrednjih zaslonov,
- opredelitev datotek oz. podatkovne baze,

(LIPP,1985) Lipp,M.E.: "Prototyping: a data-driven approach"; Pergamon Infotech Ltd., London, 1985, ss.358, s. 8/33.

(GRAD,JENKINS,1986) Grad,J., Jenkins,A.M.: "Decision Support Systems: Tools, Expectations and Realities"; osebno izročilo dr. J.Grada, 1986, ss.27.

(BOAR,1984) ibid., s.69-70.

- vključitev glavnih funkcij za interaktivno delo s podatki:
  - dodajanje novih podatkov,
  - spreminjanje obstoječih podatkov,
  - pregledovanje in listanje podatkov ter
  - brisanje podatkov.

Ugodno je, če že prvi prototip vsebuje:

* tekoči datum v glavi in možnost dela s koledarsko algebro,
* pomembne seštevke (totale),
* najbolj osnovne kontrole,
* v DO standardizirano obliko zaslonov,
* dobro opredeljene naslove zaslonov in
* najbolj osnovno sprotno pomoč na zaslonu predvsem zato, da s tem uporabniku nakažemo pomen pomoči in ga spodbudimo, da le-to dopolnjuje pri nadaljnem delu.

Mogoče ni slabo priporočilo, da je zgoraj omenjena navodila koristno uporabiti, ne glede na to ali so na začetku sploh bila dogovorjena z uporabnikom ali ne. Sploh je pri prototipnem pristopu ugodno, če imamo priložnost prijetno presenetiti uporabnika z določeno dobro rešitvijo.

Prototipni pristop predpostavlja uporabnika, ki je pripravljen za sodelovanje z oblikovalcem prototipa. Zato mu v primernem trenutku prikažemo izhodiščno različico prototipa. Oblikovalec prototipa mora biti pripravljen na običajno reagiranje uporabnika ob stiku s prototipom. Tudi če se je oblikovalec še toliko potrudil, da izdela ustrezno prespecifikacijo in prototip, bo srečanje uporabnika s prvo konkretno rešitvijo vplivalo na to, da spremeni svojo predstavo (percepcijo) o tem, kaj si dejansko želi. Njegove potrebe niso statične pač pa se vseskozi spreminjajo. Uporabnik se dejansko tudi uči ob prototipu; ideje se mu porajajo sproti - interaktivna obdelava ga animira, da dinamično daje predloge za izpopolnjevanje in dopolnjevanje prototipa. Uporabniku je treba celo omogočiti, da sam preizkuša prototip. Dobro zasnovan prototip z ustreznimi, pravilno strukturiranimi menuji za izbiro akcij, lahko usmerja uporabnika in mu olajšuje prvi stik z novim, sicer nepopolnim sistemom. Uporanik(i) in oblikovalec(i) so glavni nosilci inovativnosti pri prototipnem pristopu in ne metodologija prototipa sama po sebi.

Oblikovalec prototipa mora biti pozoren na vse pripombe uporabnika. Po potrebi lahko predebatira nejasnosti. Vsekakor si zapiše pripombe; pozneje jih bo pretehtal in verjetno vključil v novo različico prototipa. Določena programska orodja omogočajo izredno hitro delo. V kolikor korekturni poseg na prototipu ni velik, ga bo oblikovalec mogoče opravil takoj in se tako sproti prepričal, če je uporabnik zadovoljen s spremembo. Značilno za prototipni pristop je, kar potrjuje tudi mnenje J.D.Naumanna in M.A.Jenkinsa (NAUMANN,JENKINS,1982), da na ta način uporabnik posveti precej več časa svoji bodoči obdelavi, kot bi to storil pri razvoju obdelave na običajen (klasičen) način. Dejansko je uporabnik tisti, ki daje tempo razvoju prototipa.

B.H.Boar (BOAR,1984) podaja koristna priporočila, kje naj bo težišče pozornosti pri iteracijah.

Prve iteracije:

* razčiščevanje, ali je prototip v principu sprejemljiv,

* razčiščevanje, ali so vključeni vsi potrebni podatki,

* ugotavljanje, ali je prototip priročno zasnovan.

Poznejše iteracije:

* ugotavljanje, ali katera izmed potrebnih funkcij mogoče manjka oz. ali ni dobro izpeljana,

* preizkušanje različnih idej tam, kjer je to smiselno,

* prizadevanje za nadaljnjo izboljšavo celote in poenostavitev interaktivnega dela za uporabnika.

Pri spreminjanju prototipa je včasih dobro ohraniti tudi prejšnjo različico prototipa. Končno lahko tudi uporabnik občasno ugotovi, da mu prejšnja različica bolj ustreza kot nova, oz. si zamisli nova dopolnila na prejšnjo, in ne na novo različico prototipa. Dober znak, da se izdelava prototipa bliža koncu, je čedalje večja pozornost uporabnika na manj pomembne podrobnosti, kot so npr. prerasporeditev stolpcev na zaslonu ali nekoliko drugačen naziv tabele.

Praksa nam je pokazala, da je pri klasičnem pristopu, kljub potrditvi uporabnika, da se strinja s pismenimi opredelitvami bodočega sistema, težko ugotoviti ali je tudi resnično zadovoljen. To je dokaj enostavno pri prototipnem pristopu!

(NAUMANN,JENKINS,1982) Naumann,J.D., Jenkins,A.M.: "Prototyping: The New Paradigm for Systems Development"; MIS Quarterly, VI, 3, 1982, s. 29-44.
(BOAR,1984) ibid., s. 72.

v primeru večjega nezadovoljstva uporabnika je treba popolnoma spremeniti zasnovo prototipa ali mogoče bolje raziskati celotno okolje uporabnikovih potreb in računalniških možnosti; v nasprotnem primeru, ko prototip v osnovi ustreza uporabnikovim željam, imamo več možnosti za kaj in kako ga uporabiti. Te možnosti bomo razčlenili v nadaljevanju, v poglavju 4.

3. Splošni pogoji za razvoj prototipa

Po A.M.Jenkinsu (JENKINS,1983) so za idealno okolje pri prototipnem pristopu potrebni:

1. programski jezik četrte generacije (J4G) ali drugo orodje za hiter razvoj prototipa,

2. dober sistem za upravljanje podatkovne baze z enostavnim dostopom,

3. uporabnik, ki ima problem, a dobro pozna svoje področje dela in je pripravljen, da osebno sodeluje na izpopolnjevanju pototipa in

4. sposoben izdelovalec prototipa.

Zgornje pogoje bomo analizirali v nadaljevanju tako, da bomo logično združili dva pogoja, ki se nanašata na informacijsko tehnologijo in dva pogoja, ki sta subjektivne narave.

3.1. Programska orodja četrte generacije

Programski jeziki tretje generacije (COBOL, FORTRAN, BASIC, PL/I, PASCAL, ...), danes prevladujoči pri izdelavi uporabniških programov, so kljub precejšnjim medsebojnim razlikam podobni po svoji zasnovi, ki sloni na znanih von Neumannovih načelih. Z njimi izražamo operacije, ki se izvajajo zaporedno z možnostjo uporabe razvejanj in zank; so izrazito postopkovni (proceduralni).

(JENKINS,1983) Jenkins,A.M.: "Prototyping: A Methodology for the Design and Development of Application Systems"; Division of Research, School of Business, Indiana University, Bloomington, Indiana, USA 47405, Discusion Paper No. 227, 1983, ss.

J4G se medsebojno precej razlikujejo. J4G so dejansko integrirane kolekcije raznovrstne programske opreme, ki, tako povezana, omogoča določene sinergetične učinke. Nekateri jih opredeljujejo kot programske jezike visoke produktivnosti. Večkrat J4G opredeljujemo kot neproceduralne (neproceduralne) jezike kljub temu, da to ni zmeraj res. Pogosto je programsko jedro J4G postopkovno in ima le posamezne nepostopkovne možnosti. V principu postopkovni jeziki določajo KAKO se doseže določena akcija. Nasprotno pa nepostopkovni jeziki določajo KAJ se doseže, ne da bi bilo potrebno programerju opredeliti KAKO. Tako npr. v nekaterih J4G zadošča, da podamo zgolj tisto kar bi radi imeli na zaslonu, medtem ko sistem sam poskrbi za obliko in realizacijo tega. Nepostopkovne operacije pospešujejo in poenostavljajo uporabo jezika, medtem ko mu postopkovne kode dajejo dobro možnost logičnega obvladovanja. Učinkoviti J4G so vsekakor kombinacija prve in druge možnosti.

Poglejmo nekatere bistvene značilnosti J4G, ki omogočajo, da so pogosto primerni za oblikovanje prototipov:

* Prijaznost za uporabnike, ki spodbuja tudi poslovodne delavce, da sedejo pred zaslonski terminal in tudi sami komunicirajo z računalnikom. Sem pogosto sodijo: enostavnost in nedvoumnost uporabe, po potrebi sprotna pomoč neposredno ob terminalu, možnost uporabe grafike in podobno.

* Odstranitev tistih možnosti (opcij), ki lahko pripeljejo ob dvomljivi uporabi do nesporazumov ter uporaba tistih konstruktov, ki se lahko (po možnost sproti) enostavno preverjajo. Zato je večina J4G interpretativnih, kar pomeni, da programe ni treba posebej prevajati in povezovati (linkati) pred izvajanjem.

* Možnost uporabe predpostavljenih (default) opcij. Na ta način uporabniku ni treba opredeliti vse podrobnosti. Nasprotno, prevajalnik ali interpreter poda inteligentno izbrane predpostavke, ki so lahko v določenih primerih celo optimizirane.

* Enostavnost programske sintakse, ki je pri dobrih J4G razbremenjena balasta in odvečne mnemonike. Zaradi tega J. Martin ocenjuje J4G po tem, ali omogočajo uporabniku, da se jih nauči le v dveh dnevih in ga ne pozabijo po tednu dni ali več.

* Obstoj enostavnih orodij za enostavne probleme in zahtevnejših orodij za zahtevne probleme.

J4G se precej razlikujejo in zaenkrat še nimajo standarda. V grobem jih lahko ločimo glede na to ali so namenjeni predvsem izdelavi uporabniške programske opreme za upravljalce ali pa

so usmerjeni bolj transakcijsko in predvsem olajšujejo delo programerjem oz. sistemskim analitikom. Prvo skupino J4G imenujemo tudi generatorji aplikacij (angl. Application Generator Systems) in je za nas še posebej zanimiva.

Generatorji aplikacij omogočajo razvoj zahtevnih večuporabniških sistemov in so sestavljeni iz naslednjih, medsebojno povezanih delov:

- sistema za upravljanje podatkovne baze, ki je v principu relacijska ali vsaj kvazi-relacijska z možnostjo povezav na zunanje datoteke in druge podatkovne baze;

- podatkovnega slovarja, ki je popolnoma integriran z podatkovno bazo. Dejansko obvladuje podatke in njihove relacije, zaslone, programe in podprograme, odgovornosti za vzdrževanje (lastništvo) in podobno;

- generatorja transakcij in postopkovnega J4G v ožjem smislu;

- orodij za varnost in integriteto sistema;

- programske opreme za podporo pri odločanju, ki vsebuje:

  . ekstraktor podatkov,

  . preglednice,

  . grafična orodja,

  . knjižnico modelov,

  . orodja matematične statistike in drugo;

- nepostopkovnega jezika. To so:

  . jeziki za vzdrževanje podatkovne baze,

  . generatorji zaslonov in poročil,

  . vprašalni jeziki, ki omogočajo oblikovanje poizvedovanj glede na sprotne potrebe,

  . vnaprej vgrajene uporabniške funkcije za obvladovanje koledarja, finančnih izračunov in podobno,

  . možnosti za komuniciranje v naravem jeziku;

- vmesnika za povezavo osebnega računalnika v računalniško mrežo in porazdeljeno obdelavo podatkov;

- upravno-pisarniške funkcije kot so:

  . urejevalnik in besedni procesor, po možnosti s slovarjem,

  . modul elektronske pošte,

  . modul za registracijo in poizvedovanje dokumentov in podobno.

Komercialni generatorji aplikacij na osnovi pravih relacijskih podatkovnih baz so npr. ORACLE, UNIVERSE, INGRES, FOCUS in nekateri drugi. Produktivnost izdelave uporabniške programske opreme lahko povečajo kar za cel razred (do 10 krat) v primerjavi z orodji in jeziki tretje generacije.

3.2. Subjekti prototipnega pristopa

Programska orodja četrte generacije so v glavnem precej draga. Po drugi strani pa je izdelava prototipa relativno poceni, toda težavna in kreativna naloga. Zahteva oblikovalca prototipa (izvajalca), ki dobro sledi uporabnikovim zahtevam; razumeti mora celovito organizacijsko in računalniško okolje ter biti vstrajen pri izpopolnjevanju sistema.

Večina avtorjev se strinja, da je najbolje, če prototip oblikuje in sodeluje z uporabnikom le en izvajalec, ki ima lahko le svojega pomočnika. Na ta način odpade potreba po režiji in pri komuniciranju in usklajevanju ni problemov;zadostuje samo prototip, slovarj podatkov ter osebne beležke. Potreba po dodatni sprotni dokumentaciji odpade. Pri naših raziskavah prototipnih pristopov smo delali samostojno, in smo v primerjavi s klasičnimi pristopi dela v srednje velikih in velikih skupinah vseskozi imeli občutek precejšnje fleksibilnosti in možnosti za ekspeditivnost, kljub obveznostim v drugih nalogah. Po naših izkušnjah je ugodno, da je oblikovalec prototipa ekstravertne narave, komunikativen in pripravljen za tesno, prijateljsko sodelovanje z uporabnikom.

Prototipni pristop je uresničljiv le tam, kjer ima uporabnik dosti časa in motivacije za skupno delo in izpopolnjevanje prototipa. Seveda mora biti seznanjen s to informacijsko tehnologijo. Pri našem delu smo bili v stiku z uporabnikom na dnevni osnovi. Večinoma so zadostovale le kratke izmenjave mnenj, pogosto le po telefonu. Vsakih tri do štiri dni pa smo

skupaj preizkušali izpopolnjene prototipe. V literaturi nismo našli opise primerov, pri katerih bi na strani uporabnikov bilo več aktivnih preizkuševalcev prototipa. Pri nas se je v enem primeru harmoničnega sodelovanja izkazala tudi ta možnost, saj po stari resnici, da več oči več vidi in več glav več ve.

Še enkrat poudarjamo, da je izdelava prototipov kreativen proces, tako za oblikovalce prototipov kot tudi za uporabnike. Zato je zadovoljstvo toliko večje, ko se končno pride do ustreznega prototipa. Izkušnje potrjujejo dejstvo, da prototip ne samo omogoča, ampak celo spodbuja uporabnika, da spreminja svoje želje glede na svoje potrebe. Ravno občutek, da le-ta spreminja svojo percepcijo in se orientira upravljalno in ne le operativno-izvajalno, in to že v zgodnji fazi razvoja, daje prototipu tisto moč, ki jo rabimo pri razvoju računalniško podprtih upravljalnih informacijskih sistemov.

4. Možnosti za uporabo računalniških prototipov

Strokovna literatura sicer navaja za kaj se uporablja računalniški prototip, toda v glavnem ne razčlenjuje na kakšen način in od česa je odvisen način uporabe.

Prototip lahko uporabimo predvsem kot:

1. fazo v življenjskem ciklusu razvoja računalniško podprtega informacijskega sistema kot sistemsko-analitično orodje;

2. izhodišče za izpopolnitev prototipa v obdelavo za redno uporabo s pomočjo običajnih programskih orodij;

3. delujoči sistem za potrebe upravljalnega informacijskega sistema, zlasti za improvizirano poročanje in analize ter za podporo pri procesu odločanja;

4. pilotski izdelek oz. prototip v klasičnem pomenu besede.

V nadaljevanju bomo obravnavali le prve tri možnosti. M.A.Janson in D.L.Smith sta primerjala "klasične" materialne (inženirske) prototipe z informacijskimi (JANSON,SMITH,1985) in ugotovila, da se prvi lahko uporabljajo za:

(JANSON,SMITH,1985) Janson,M.A., Smith,D.L.: "Prototyping For Systems Development: A Critical Appraisal"; MIS Quarterly, IX, 4, 1985, s. 305-316.

1. preverjanje, če so uporabnikove (naročnikove) zahteve zadovoljene;

2. preverjanje, če se zasnova lahko specificira;

3. izbiro najboljše zasnove izmed več prototipov;

4. celovito preizkušanje, ki sloni na novi rešitvi s prototipom;

5. celovito preizkušanje v različnih okoljih uporabe;

6. predstavitev odločevalcem v smislu potrditve ali zavrnitve nadaljnjega razvoja;

7. uvajanje novega sistema v uporabnikovo okolje.

Skoraj vse zgornje možnosti uporabe pridejo v poštev tudi pri računalniških prototipih.

Pomembno je, da opredelimo še razliko med računalniškim prototipom in ustreznim končnim izdelkom, pripravljenim za redno obdelavo. Po C.Floydu (FLOYD,1984) je ta razlika značilna, kajti drugače se prototip in končni izdelek ne bi razlikovala. Razlika izhaja predvsem iz določene funkcijske omejitve prototipa. Ločimo

1. VERTIKALNI PROTOTIP, ki pa ponuja le določene izbrane funkcije bodoče celote v končni obliki in

2. HORIZONTALNI PROTOTIP, ki sicer ima vse funkcije, ki jih potrebuje končni izdelek, toda le-te so realizirane samo delno, tako da so podrobnosti izpuščene ali simulirane.

4.1. Prototipni pristop kot sistemsko analitično orodje

Medtem ko večina avtorjev meni, da je računalniške prototipe možno uporabiti na različne načine, je B.H.Boar dosleden v

---

(FLOYD,1984) Floyd,C.: "A Systematic Look at Prototyping"; s. 1-18 v zborniku Approaches to Prototyping: Proceedings of the Working Conference on Prototyping, Namur, October 1983; Springer Verlag, Berlin, 1984, ss. 458.

(BOAR,1984) ibid.

(BOAR,1986) Boar,B.H.: "Application Prototyping: A Life Cycle Perspective"; Journ. of Sys. Management, XXXVII, 2, 1986, s. 25-31.

prepričanju, (BOAR,1984) in (BOAR,1986), da je računalniške prototipe možno uporabiti zgolj kot dobra sistemsko analitična orodja za opredelitev (definicijo) potreb v fazi informacijske analize potreb (gl. fazo 2. na str. 1) življenjskega ciklusa razvoja sistema.

Naša teza je, da je možna namembnost uporabe prototipov močno pogojena z velikostjo in hitrostjo razpoložljivega računalnika kakor, tudi z vrsto in bogastvom programske opreme četrte generacije, ki je na razpolago pri razvoju prototipov. B.H.Boar je do svojih izkušenj najbrž prišel ob enostranski uporabi programske opreme IDMS softwarskega proizvajalca Cullinet. Ne glede na to, je B.H.Boar ustrezno opisal splošne značilnosti uporabe prototipnega pristopa kot sistemsko analitičnega orodja, zato bomo prevzeli predvsem njegove misli.

Zmotna so mnenja, ki preveč idealizirajo in poenostavljajo prototipni pristop (ponavadi to počno tisti, ki niso delali na takšen način) v smislu:

* uporabnik in oblikovalec prototipa pregledata potrebe (zahteve),

* oblikovalec takorekoč sproti kreira prvo različico prototipa,

* po prvi preizkušnji se takoj odpravijo pomanjkljivosti,

* uporabniku se prototip dopade in uporaba prototipa se začne kot redna ali občasna obdelava,

* problemov ni več in vsi so zadovoljni.

Zanimivo je, da se pri razvoju inženirskih prototipov ne postavlja vprašanje neposredne uporabe prototipov. Znano je pač, da je treba opraviti še precej dela t.i. proizvodnega razvoja. Enako pravilo naj bi veljalo tudi pri računalniških prototipih. Obsežna dela, ki zanesljivo ostanejo, so opredelitev (oz. uresničitev):

1. sistemskih vhodov in izhodov:

   - obsega obdelave in lokacije terminalskih mest,

   - frekvence obdelav,

   - varnosti podatkovne baze in obdelave ter

   - ukrepov v primeru zastojev;

2. vseh podrobnosti v zvezi s podatkovno bazo;

3. konverzijskega postopka, kako in odkod se bo napolnila podatkovna baza;

4. podrobne sistemske logike in zlasti vključitev vseh kontrol, kot so:

   - kontrola podatkov na numeričnost, dolžino in območje vrednosti,

   - kontrola podatkov v odvisnosti od vrednosti določenih drugih podatkov,

   - kontrole ključev, opredeljene logike, datumov in druge kontrole;

5. zanesljivosti sistema, kot so pretočnost podatkov, odzivni časi,...;

Vse zgoraj naštete potrebe v bistvu sodijo v podatkovni slovar. Vsekakor pa je treba oceniti, kakšen je vpliv teh zahtev na prototip. Znano je dejstvo, da programska orodja četrte in pete generacije sicer nudijo marsikatere prednosti, a so na računalniškem prostoru in času precej potratna. V kolikor analiza in ovrednotenje zmogljivosti pokaže, da le-te ne zadoščajo, nadaljujemo z ostalimi fazami življenjskega ciklusa razvoja obdelave na običajen način in z običajnimi programskimi orodji. V strokovni literaturi je opisanih nekaj primerov, kjer so začeli uporabljati prototip kot redno obdelavo, ne da bi fizično razrešili vrsto zgornjih vprašanj: posledice so bile katastrofalne.

Logične strukture, ki so primerne za izdelavo prototipov v smislu sistemsko analitičnih orodij, so strukturirani problemi s precejšnjo množico objektov in atributov v datotečnih povezavah, toda z relativno majhnim deležem algoritmičnih procesov. Dobri kandidati so torej transakcijski sistemi operativne narave. Obdelave, namenjene odločevalcem, niso primerne. Na splošno pa velja, da paketne (batch) obdelave in celotne informacijske sisteme ne smemo razvijati s prototipnim pristopom, kar bomo razložili pozneje.

4.2. Predelava prototipa v obdelavo za redno uporabo

Predelava prototipa v obdelavo za redno uporabo je dokaj pogosta, toda po našem mnenju v bistvu ne prinaša večjih kakovostnih razlik glede na način uporabe, ki je opisan v podpoglavju 4.1. Značilno za oba pristopa je, da je

računalniško obdelavo potrebno ponovno kodirati in testirati, toda tokrat z izbranim postopkovnim jezikom (običajno COBOL) in standardnimi pripomočki v določenem okolju. V čem je torej razlika? Predvsem v tem, kako podrobno in zanesljivo lahko opredelimo bodoči sistem s prototipnim pristopom in koliko dela nam ostane, da razčistimo s klasično sistemsko analizo. V primeru, ki ga obravnavamo, se večina analiz in končna predstavitev uporabniku izvede preko prototipa. Ta način je zato bolj primeren za nekoliko manjše obdelave, kajti na vse, v prejšnjem podpoglavju naštete zahteve, težko odgovorimo pri veliki obdelavi zgolj s preizkušanjem na prototipu.

Pri obeh opisanih vrstah uporabe prototipov so v praksi manjši efekti na pridobljenem času in neprimerno večji na kakovosti in zanesljivosti opredelitve uporabnikovih potreb. Splošno priznanih, teoretično opredeljenih in praktično dokazanih vrednotenj obeh sistemov še ni. Nagibamo se k lastnemu izkustvenemu mnenju, da se oba pristopa splača uporabiti zgolj v primerih, ko uporabnik ne zna dobro opredeliti svoje potrebe, a ima zadosti dobre volje in časa za sodelovanje z oblikovalcem prototipa.

4.3. Uporaba prototipov za potrebe upravljalnih informacijskih sistemov

J.D.Naumann in A.M.Jenkins menita (NAUMANN,JENKINS,1982), da so za prototipni pristop izgradnje računalniške obdelave najbolj primerne poslovne funkcije vodilne strukture v organizaciji. Podobnega mnenja so še C.B.McNurlin (McNURLIN,1981), R.H.Reck, J.R.Hall (RECK,HALL,1986) in nekateri drugi.

Uporaba prototipov za potrebe upravljalcev sloni predvsem na prototipih, ki se z računalniškimi orodji četrte generacije pripravijo z dodatnim delom za uporabno obdelavo. Pri tem je treba razrešiti vprašanja, ki smo jih nakazali v podpoglavju 4.1. R.C.Cullum (CULLUM,1985) navaja še šest vprašanj, ki se nanašajo na uporabo prototipov kot rednih ali občasnih obdelav. Samo pritrdilen odgovor na ta vprašanja v podjetju je porok, da bo takšen pristop uspešen v praksi. Vprašanja

---

(NAUMANN,JENKINS,1982) ibid.
(McNURLIN,1981) McNurlin,B.C.: 'Developing Systems by Prototyping'; EDP Analizer, XIX, 9, 1981, s. 1-12.
(RECK,HALL,1986) Reck,R.H., Hall,J.R.: 'Executive Information Systems: An Overview of Development'; Journal of Information Systems Management, III, 4, 1986, s. 25-30.
(CULLUM,1985) Cullum,R.L.: 'Iterative Development'; Datamation, XXXI, 4, 1985, s. 92-98.

so:

1. Ali je poslovodna struktura pripravljena reševati informacijske potrebe po kosih?

2. Ali je poslovodna struktura pripravljena na to, da zavrže prototip, če le-ta ne ustreza?

3. Ali oblikovalec prototipa dobro pozna računalniška orodja četrte generacije, ki so mu na razpolago?

4. Ali so zagotovljene zadostne računalniške zmogljivosti?

5. Ali se poslovodna struktura zaveda rizika glede uspešnosti rešitve?

6. Ali so oblikovalci prototipa pripravljeni pozneje izdelati popolno dokumentacijo?

Mnenja smo, da je izdelava obdelav za upravljalce s prototipnim pristopom smiselna le, če imamo možnost razviti vsebinsko zadovoljiv prototip v uporabno obdelavo z istimi orodji četrte generacije, s katerimi smo razvili prototip. To je uresničljivo samo, če razpolagamo z ustreznim računalnikom in ustrezno, zadosti bogato programsko opremo - s kakovostnimi generatorji aplikacij.

Prepričani smo, da so relacijske podatkovne baze (RPB) najbolj primerne za izdelavo prototipov in končnih obdelav za upravljalce. J.Connell in L.Brice (CONNELL,BRICE,1984) sta realizirala več kot 20 prototipov v okolju RPB in menita, da je zares učinkovit prototip mogoč le s pravimi, sodobnimi RPB. Po C.J.Datu (DATE,I,1986) je bilo na svetu l. 1986 okoli 25 različnih pravih RPB. Približno polovica je bila SQL-tipa (ORACLE, DB2, SQL/DS UNIFY, ...), ostala polovica pa drugačnih (INGRES, CA-UNIVERSE, dBASE III, NOMAD, ...). V prodaji je precej kvazi-relacijskih podatkovnih baz, ki jih proizvajalci reklamirajo kot RPB. To so le relacijski vmesniki na druge upravljalne sisteme podatkovnih baz (npr. ULTRA, proizvajalca CINCOM Systems).

Relacijske podatkovne baze so računalniško dokaj potratne, toda omogočajo hitro spreminjanje shranjene strukture,

---

(CONNELL,BRICE,1984) Connell,J., Brice,L.: "Rapid Prototyping", Datamation, XXX, 13, 1984, s. 93-100.

(DATE,I,1986) Date,C.J.: "An Introduction to Database Systems, Vol.I"; Addison-Wesley Publ. Comp., Reading, Mass., 1986, ss.639, s. 325.

(BOTTOM et al.,1985) Bottom,J., Bernard,A., Anderson,K.: "The Art of Modeling"; Datamation, XXXI, 22, 1985, s. 140-146.

neodvisno od fizičnega upravljanja s podatki (verige, kazalci, ključi, ...). Bojazni, da se bodo shranjeni podatki izgubili, ni. Imajo tudi vse potrebne atribute za redno uporabo, kot so npr. orodja za varnost idr.

Primerno računalniško okolje za razvoj obdelave za upravljalca v poslovnem sistemu je mikroračunalnik v povezavi z osrednjim računalnikom. Osebni računalnik ponuja poslovodnemu organu še večjo varnost in privatnost podatkov. Povezava omogoča lokalne obdelave z ekstrahiranimi podatki iz poslovnih datotek. J.Bottom, A.Bernard in K.Anderson (BOTTOM et al.,1985) ugotavljajo, da že obstaja zadovoljiva programska oprema za razvoj prototipov na osebnih računalnikih.

5. Izkušnje s prototipnim pristopom

Izdelali smo nekaj prototipov. Iz študijskih razlogov smo bili pri delu posebno pozorni na:

1. vsebino informacijskih potreb,

2. vse aspekte prototipnega pristopa in zlasti na

3. ustrezno informacijsko tehnologijo četrte generacije.

Na tem mestu ne bomo obravnavali vsebinske aspekte. Težišče bo na drugih dveh momentih. Podali bomo tudi nekatere tuje izkušnje. Iz praktičnih razlogov bomo obravnavo zožili in jo uredili po programskih orodjih. Izkušnje s programskim jezikom PROLOG v članku ne bomo opisali.

5.1. Programska orodja System 1022 in System 1032

S Systemom 1022, programsko opremo četrte generacije (proizvajalec Software House, ZDA), ki je postavljena na računalnikih DEC-10 in/ali DEC-20 v Računalniškem centru Univerze v Ljubljani, smo naredili dva večja prototipa:

1. prototip pomožnega priročnega informacijskega podsistema za delo v izvoznih oddelkih Iskre in

2. prototip pomožnega priročnega informacijskega podsistema za spremljanje izvajanja obveznosti na samoupravni in poslovodni ravni Iskre.

Prvi prototip smo razvijali samostojno tri mesece in drugega dva tedna, v povprečju po 3,5 ure na dan. Skupina avtorjev (LANGLE et al.,1984) je l. 1981 ugotovila, da so takrat v ZDA le redka podjetja uporabljala prototipni pristop, in da so prototipe razvijali od tedna do tri mesece.

Kljub temu, da smo prvič delali s Systemom 1022 in s prototipnim pristopom, smo izdelali zadovoljiv prototip približno trikrat hitreje, kot če bi bil enak prototip narejen s programskim jezikom COBOL. To sovpada z rezultati, ki sta jih dobila E.C.Harel in E.R.McLean (HAREL,McLEAN,1985) s primerjavo znanega J4G FOCUS in najbolj razširjenega J3G COBOL. Zelo izkušeni COBOL programerji so prototipe izdelali približno dvakrat počasneje od FOCUS programerjev.

J.Martin je ugotovil (MARTIN,1983), da ima program napisan z jezikom COBOL, 20 krat do 40 krat več kodirnih vrstic (ena kodirna vrstica stane v ZDA v povprečju 10$!) kot vsebinsko enak program, napisan z J4G. Naše izkušnje se gibljejo na spodnji meji J.Martinove ocene, s pripombo, da bi bila izvedba z jezikom COBOL že profesionalna, medtem ko bi rešitev s Systemom 1022 ostala na meji prototipne improvizacije, vsaj kar se tiče performans za uporabo.

System 1032 je različica, podobna Systemu 1022, istega proizvajalca, prilagojena za računalnike VAX in operacijski sistem VMS. Za razliko od Systema 1022 leta 1986 še ni obstajala možnost neposredne povezave na paket matematične statistike SPSSX; drugače ni bilo večjih razlik.

System 1022 in System 1032 sta pisana izključno za računalnike firme DEC (Digital Equipment Corporation), zato je navezanost na operacijska sistema TOPS oz. VMS izredno močna. To sicer izboljšuje zmogljivosti sistema, po drugi strani pa onemogoča prenos na druge vrste računalnikov. Opis podatkovne baze in podatke imata sistema zakodirana na poseben način. Upravljalna sistema podatkovne baze sta v bistvu mrežnega tipa, z določenimi kvazi-relacijskimi značilnostmi. Nepostopkovni jezik je učinkovit in enostaven, postopkovni pa dokaj neroden. Možna je povezava z jezikom COBOL. Manjka dober oblikovalec zaslona, medtem ko je avtomatični generator poročil oz. pripomoček za samostojno oblikovanje poročil

---

(LANGLE et al.,1984) Langle,G.B., Leitheiser,R.L., Naumann,J.D.: "A Survey of Applications Systems Prototyping in Industry"; Information and Management, VII, 5, 1984, s. 273-284.

(HAREL,McLEAN,1985) Harel,E.C., McLean,E.R.: "The Effects of Using a Nonprocedural Computer Language on Programmer Productivity"; MIS Quarterly, IX, 2, 1985, s. 109-120.

(MARTIN,1983) Martin,J.: "Software for Application Development Without Conventional Programming"; Software World, XIV, 1, 1983, s. 14-21.

povprečen. Dobri lastnosti sistemov sta vključenost koledarjev za približno 100 let in odlična koledarska algebra. Zato sta oba naša prototipa slonela na učinkovitem obvladovanju zahtevnih terminskih obveznosti.

Glede na ceno (približno 40.000 $) in na to, da sistem ne sloni na relacijski podatkovni bazi, odsvetujemo uporabo Systema 1032 v Iskri. Kljub temu zapišimo, da je oba sistema možno uporabljati za izdelavo prototipov, ki se lahko izpopolnijo do uporabne obdelave. Primera tovrstnih obdelav sta v Splošni plovbi Piran - na Systemu 1022 in v Novolesu, Novo Mesto - na Systemu 1032.

5.2. Programska orodja CA-UNIVERSE

Programska oprema četrte generacije CA-UNIVERSE je novejši proizvod ameriške firme Computer Associates International, Inc. iz Jericha, NY. Uporabna je za končne uporabnike, kot tudi za računalniške strokovnjake - oblikovalce prototipov in obdelav. Prirejena je za velike IBM računalnike (16 MBy centralnega spomina in več). Izvaja se lahko pod operacijskimi sistemi VSE, VS1, MVS ali VM/CMS.

CA-UNIVERSE je pravi relacijski upravljalni sistem podatkovne baze QUEL-tipa z vrsto drugih orodij. Ima popolnoma integriran podatkovni slovar, relacijske funkcije JOIN, PROJECT, SELECT, generator zaslonov, generator poročil, sortne funkcije in funkcije, ki skrbijo za varnost in integriteto sistema. Vsebuje tudi relacijsko zasnovan vprašalni jezik in postopkovni jezik imenovan ADL. Za interpretacijo visoko nivojskih relacijskih CA-UNIVERSE ukazov, ki se lahko vgnezdijo bodisi v COBOL, PL/I, FORTRAN ali asemblerske programe, uporablja predprocesor.

Ima dve ravni generatorja zaslonov in generatorja poročil: z enostavnim pristopom za končne uporabnike in z raznovrstnimi možnostmi za strokovnjake. Generator poročil za strokovnjake se imenuje CA-EARL.

Sistem je po našem mnenju idealen za izdelavo prototipov od izhodiščnega prototipa do končne obdelave, primerne za upravljalce. Sistem omogoča popolno rezervno varnost (backup) ali ponovni start (restart) in funkcije nadziranja na osnovi dnevnika (log file). Za oblikovalca prototipa je CA-UNIVERSE izredno primeren in ekspeditiven, ker se le-ta z ukazom Q (QUIT) lahko neposredno prestavi iz enega področja dela v drugo (in po potrebi tudi nazaj).

Pomembno je, da sistem v povezavi z CA-EXECUTIVE omogoča ekstrakcijo podatkov s centralnega računalnika v osebni

računalnik in nadaljnjo lokalno obdelavo teh podatkov, npr. s preslednico. Lokalna oprema vsebuje še grafična orodja, urejevalnik in besedni procesor z odličnim slovarjem za preverjanje pravilnosti angleških besed (word-checker) s približno 88.000 angleških besed, kar je idealno orodje za pisanje dopisov in drugih materialov v angleškem jeziku prav v našem okolju. Preprišali smo se v učinkovitost teh orodij, kakor tudi v izredno nazornost večbarvne tridimenzionalne grafike v povezavi s preslednico.

CA-UNIVERSE in CA-EXECUTIVE uporabljajo v DO Iskra Telematika in Slovenija ceste tehnika približno dve leti. V DO Telematika imajo z CA-UNIVERSE sprogramirano planiranje in spremljanje posebnega dela proizvodnje namenjene izvozu za IBM v ZR Nemčiji. Obdelava je bila razvita prototipno. V podatkovni bazi je bilo v prvi polovici januarja 1987 opredeljenih 25 relacij (tabel), kot so npr. tehnološki plan (koda materiala, koda proizvodne operacije, norma ure, ...), naročila (št. naročila, koda materiala, količina, ...) in druge.

Osrednja pomanjkljivost prototipnega razvoja v DO Telematika je ta, da niso pravočasno opravili normalizacijo podatkov. Tako npr. še danes vlečejo breme tega, da niso upoštevali logično odvisnost možnih zastojev v proizvodnji tudi od pomanjkanja materialov (od kode materialov). Kljub temu lahko ocenimo obdelavo kot uspešno. Razvila sta jo oblikovalec in predstavnik uporabnikov. Obdelava se izvaja preko štirih terminalov. Računalniško je dokaj potratna, toda odzivni časi so v glavnem sprejemljivi. V principu bi se lahko rezultati pošiljali neposredno k IBM po računalniški mreži, toda v praksi se elektronsko prenašajo natiskane tabele kot faksimile.

5.3. Programska orodja MANTIS

MANTIS, proizvod firme Cincom Systems, Inc. iz ZDA, ponuja vrsto povezanih orodij četrte generacije, ki se dosledno izbirajo na osnovi menujev. Na izbiro so oblikovalci zaslonov in datotek, interpreterski postopkovni jezik, ki omogoča modularno in strukturno programiranje, možnosti za izdelavo

---

(MARTIN,II,1986) Martin,J.: "Fourth-generation Languages, Vol.II, Representative 4GLS"; Prentice-Hall, Englewood Cliffs, NJ., 1986, ss. 500, s. 247.

(MANTIS,P25-1210,1985) -: "MANTIS Prototyping Supplement"; Cincom Systems, Inc., Cincinnati, OH., P25-1210, 1985, ss. 64, s. I-5.

(FRANK,1984) Frank,W.L.: "Over-friendly Software"; Data processing, XXVI, 4, 1984, s. 28-30.

scenarijev idr.

Po mnenju J.Martina (MARTIN,II,1986) MANTIS zaostaja za nekaterimi drugimi bogatimi orodji četrte generacije, predvsem zato ker nima:

1. svoj poizvedovalni jezik,

2. generatorja poročil,

3. popolni podatkovni slovar,

4. vmesnika za komuniciranje v naravnem (angleškem) jeziku ter

5. enostavne vmesnike za izkoriščanje razpoložljivih možnosti.

Toda po zagotovilu J.Martina se Cincom močno prizadeva, da bi odpravil te pomanjkljivosti.

Osrednja orodja, ki jih ponuja MANTIS za izdelavo prototipov (MANTIS,P25-1210,1985) so:

- MANTIS postopkovni programski J4G,

- sistem za izvedbo programov in

- pripomočki za izdelavo scenarijev.

S stališča oblikovalca prototipa je MANTIS (po našem mnenju) sprejemljivo orodje le za hitro opredelitev sistemsko analitičnih potreb srednje velikih operativnih informacijskih podsistemov. Specifična, toda zares uporabna orodja, so pripomočki za izdelavo scenarijev, zlasti za manj izkušene oblikovalce. Za izkušene oblikovalce prototipov je precej nepraktična in dolgočasna potreba, da se nenehno "sprehaja" preko vseh hierarhičnih nivojev menujev, brez možnosti preskokov, na osnovi posebnih ukazov. Manjka tudi dobra koledarska algebra. Oboje pa je po splošnem mnenju W.L.Franka (FRANK,1984) dokaj velika pomanjkljivost.

Postopkovni programski jezik MANTIS je zmogljiv, toda zahteven. Kot zanimivost naj povemo, da je tudi celotna programska oprema MANTIS sprogramirana z jezikom MANTIS, ki je izrazito interpreterski jezik, a kljub temu relativno dobro optimiran. Pomembna lastnost MANTISA je, da se lahko izvaja tako na računalnikih IBM kot tudi na računalnikih tipa VAX (proizvajalca DEC iz ZDA). Sodeč po (dobrih) priročnikih, je ta prenos z računalnika na računalnik dokaj enostaven. Preizkusili ga nismo.

MANTIS je v svetu precej popularen. Tudi v Sloveniji se občasno uporablja v Iskra CAOP, Iskra Commerce, v Ljubljanski banki - Gospodarski banki, na Prometnem institutu in drugod. Zdi se nam umestno še nekoliko počakati na dopolnitve v novih izdajah MANTISA, pred začetkom masovnejše uporabe tega orodja.

6. Povzetek značilnosti uporabe prototipnih pristopov: koristi in pasti

Nekatere bistvene prednosti prototipnega pristopa lahko vidimo iz primerjalne tabele, ki smo jo deloma priredili po (STRELAU,1984).

PRIMERJALNA TABELA ZNAČILNOSTI

TRADICIONALNI RAZVOJNI CIKLUS	RAZVOJNI CIKLUS NA OSNOVI PROTOTIPA
-predstavitev s svinčnikom in papirjem	-fizična realizacija sistema
-trdne pogodbe o zasnovi z uporabnikom	-heurističen in kreativen razvoj
-daljši časi razvoja	-krajši časi razvoja
-slaba odzivnost na spremembe	-večja funkcionalnost in fleksibilnost
-uporabnik ne obvladuje razvoj	-uporabnik obvladuje razvoj

(STRELAU,1984) Strelau,F.: "System Prototyping"; str. 1/2/1-1/2/20 v State of the Art Review: the European Challenge; Stream 1, Pergamon Infotech Ltd., London, 1984.

(KRSTIC,1986) Krstič,D.: "Računalniške preglednice: nov programski pripomoček v poslovnem informacijskem sistemu"; Ekonomska revija, XXXVII, 2, 1986, s.183-193.

(BARLE,GRAD,KRSTIC,1986) Barle,J., Grad,J., Krstič,D.: "A Production Problem Interactive Prototype of Linear Programme"; Referat na srečanju Deutsche Gesellschaft fur Operations Research, Ulm, september 1986, ss. 8.

Na osnovi naših izkušenj je prototipni pristop še posebej opravičen, če gre za izdelavo obdelav za upravljalce. Seveda sè za to potrebujejo kakovostni generatorji aplikacij z relacijskimi podatkovnimi bazami in zmogljivi računalniki. Zaželena je uporaba čimbolj raznovrstnih orodij, primernih za podporo pri odločanju. V te namene se lahko učinkovito uporabijo tudi računalniške preglednice(KRSTIC,1986) ali modeli na osnovi linearnih programov (BARLE,GRAD,KRSTIC,1986).

Z uporabo prototipov, ki se razvijejo z enimi orodji skupaj z uporabnikom, in se potem preprogramirajo v uporabno obdelavo z drugimi orodji, nismo zadovoljni. Izkušnje v Tozd CAOP so pokazale, da različna programska logika različnih orodij precej moti projektanta obdelave. Zdi se nam, da je v teh primerih bolje za projektnta pripraviti klasično sistemsko analitično dokumentacijo.

Da se izognemo nerodnostim, opisanimi v zgornjem odstavku, smo v Tozd CAOP razvili določen hibridni prototipni pristop. Kadar v določenem okolju obstojajo bogate kolektivne izkušnje in bogate programske knjižnice, je možno izvajati prototipe tudi s postopkovnim jezikom, kot je COBOL. Razen uporabnika je možno vključiti v vsebinske in programske iteracije tudi širši krog sistemskih analitikov, projektantov in programerjev; toliko bolj, če gre v bistvu za modularno povezane obdelave različnih področij. Na tak način je tudi določene lokalne standarde lažje vzdrževati.

Za paketne obdelave interaktivni dialog ni bistven, zato prototipni razvoj ne pride v poštev.

Končno ponovimo opozorilo: izdelava prototipa celotnega informacijskega sistema zaradi obsežnostini mogoča (vsaj za enkrat), in je kot taka, obsojena na neuspeh. Obljube v tej smeri lahko pozneje povzročijo izredno velike poslovne traume in nepopravljivo škodo računalniški klimi v organizaciji. Tovrstna zavajanja niso redka. Žalostno je, da jih "priporočajo" celo nekateri svetovalci renomiranih slovenskih svetovalnih organizacij. Tisti, ki je delal s prototipi to zagotovo ne bo počel. Zato, če ste uporabnik, zahtevajte od svetovalca, naj vam pokaže svoje prototipe. Ravno v tem je velika prednost prototipov, da si jih lahko ogledamo "v živo".

mag. Andrej KOVAČIČ

DO PRIS,
Projektiranje informacijskih
sistemov in inženiring

Ljubljana

STRATEGIJA RAZVOJA INFORMACIJSKEGA SISTEMA ZA PODPORO ODLOČANJU

1 UVODNE OPREDELITVE

Sodoben, računalniško zasnovan informacijski sistem naj, razen obravnave podatkov operativnih funkcij na transakcijskem nivoju, predvsem zagotavlja ustrezne podatke za pridobivanje informacij za podporo odločanju na nadzornem in upravljalnem nivoju poslovnega sistema. Zadovoljevati mora trenutne in bodoče informacijske potrebe uporabnikov.

Tako opredeljene cilje lahko dosežemo le z ustreznim sodobnim pristopom h gradnji informacijskega sistema. Podatkovna zasnova, strateško načrtovanje in prototipni pristop ob uporabi sodobnih informacijskih orodij zagotavljajo postopno gradnjo s sprotnim preverjanjem ustreznosti rešitve ter zahtevano fleksibilnost in prilagodljivost informacijskega sistema spremembam poslovnega sistema in okolja.

V vseh fazah gradnje informacijskega sistema je nujno aktivno vključevanje uporabnikov v skupne projektne time. Tehnološko znanje uporabnikov, ob novo pridobljenem informacijskem znanju, zagotavlja izgradnjo informacijskega sistema, ki bo dolgoročno pokrival informacijske potrebe poslovnega sistema.

Področje odločanja predstavlja eno ključnih dimenzij upravljanja, ki zajema ugotavljanje problemov, vzrokov za njihov nastanek, predvidevanje možnih rešitev ter izbiro in uveljavljanje najustreznejše rešitve. Le optimalne odločitve pa vodijo k ciljnemu delovanju poslovnega sistema oziroma k njegovi učinkovitosti, donosnosti in zadovoljevanju potreb zaposlenih.

Značilnost odločanja se kaže v dejstvu, da v trenutku odločitve bodoče posledice lahko le predvidimo. Čim težje so posledice odločitve, tem večjo vrednost imajo informacije, ki jih lahko zagotavlja informacijski sistem. Le-ta mora na področju odločanja izpolniti predvsem pogoj enostavne in učinkovite komunikacije z uporabnikom-odločevalcem, obremenjenim s kompleksnostjo informacijskih orodij, in obsežno ter pestro strukturirano količino podatkov, potrebno za kakovostno odločanje.

2 SODOBNI PRISTOPI GRADNJE INFORMACIJSKEGA SISTEMA ZA PODPORO ODLOČANJU

Tradicionalni pristopi razvoja informacijskih sistemov se običajno osredotočajo na posamezne aplikativne segmente in na obravnavo posameznih postopkov poslovanja, kar vodi k ločenemu razvoju operativno neodvisnih uporabniških programskih rešitev. Podatkovne strukture nastajajo kot dodaten, vzporeden proizvod razvoja programov, kar povzroča probleme podvajanja in necelovitosti podatkov. Z rastočim številom uporabniških rešitev, ki pokrivajo le operativni nivo funkcij poslovnega sistema, se veča kompleksnost in togost informacijskega sistema, saj so spremembe programske opreme izredno drage in zámudne. "Obdelava podatkov" ni sposobna zagotavljati podatkov za pridobivanje informacij za potrebe odločanja.

Spoznanja, da so podatki eden bistvenih virov vsakega poslovnega sistema, pojav krmilnih sistemov baz podatkov in pojav novih tehnoloških možnosti (informacijska orodja, "močni" osebni računalniki, omrežja za prenos podatkov...), so vodila do poskusov formalnega opredeljevanja in organiziranja podatkovnih struktur v bazah podatkov. Vendar razvoj uporabniških programskih rešitev, ki še naprej temelji na filozofiji postopkovno orientiranih pristopov k izgradnji informacijskih sistemov in klasičnih, postopkovno usmerjenih programskih jezikih, zahteva zamudno načrtovanje, programiranje in testiranje teh rešitev.

Da je problem v praksi očiten, nam kaže množica neuspešno zasnovanih baz podatkov in uporaba sodobnih informacijskih orodij, ki razen višjih stroškov delovanja informacijskega sistema ne prinaša nikakršnih prednosti pred klasičnimi "obdelavami" računskih centrov (AOP). Zaradi upravičenih informacijskih potreb uporabnikov (posebno odločevalcev) pa se širi neusklajena uporaba nepovezanih osebnih računalnikov, kakor tudi zmotno razmišljanje informatikov o ločenih in različnih pristopih h gradnji informacijskih sistemov operativne in odločitvene (strateške) ravni.

Če smo se v preteklosti še lahko slepili z ugotovitvami, da je gradnja računalniško zasnovanega informacijskega sistema predvsem tehnološki problem, sedaj ugotavljamo, da neverjetne rasti kompleksnosti, ki jo povzročahitro rastoče informacijske potrebe uporabnikov, z obstoječimi metodami le-teh nismo več sposobni obvladovati. Ugotovitev velja tako na operativnem še posebej pa na strateškem nivoju.

Nove metodologije gradnje informacijskih sistemov, ki vključujejo nove tehnološke možnosti, pogojujejo drugačno

organizacijsko, ekonomsko pa tudi vse bolj sociološko gledanje na to problematiko. Usmerjajo se v ustroj, postopke, ciljne usmeritve in strategijo organizacije. Ključni cilj zasnove novega informacijskega sistema predstavlja učinkovitost celotne organizacije (medtem, ko se klasični metodološki pristopi ukvarjajo z učinkovitostjo posameznih funkcij in aktivnosti). Učinkovitost je sicer merilo stopnje doseganja ciljev, ki pa jih moramo najprej opredeliti na različnih ciljnih nivojih. Vsebinsko doseganje ciljev nižjih nivojev ocenjujemo s stališča podpore ciljem višjih nivojev. Sodobne metodologije izhajajo na tem področju iz naslednjih predpostavk in aktivnosti:

- ugotavljanje združljivosti ciljev temeljnih poslovnih funkcij s stališča usmerjenosti k skupnim ciljem organizacije,

- ugotavljanje niza poslovnih pravil, meril in pripadajočih informacij pomembnih s stališča učinkovitosti organizacije,

- ugotavljanje metodoloških zahtev zasnove novega informacijskega sistema potrebnih za uskladitev informacijskih zahtev z značilnostmi organizacije in upravljalnih struktur.

Namesto vprašanja, ki je bilo največkrat prisotno in se je glasilo: "S kakšnim postopkom (uporabniškim programom) bomo reševali problem?" se vse bolj uveljavlja vprašanje: "Kateri podatki bodo zadostili trenutnim in bodočim informacijskim potrebam organizacije?". Iz vprašanja izhaja, da mora organizacija ugotoviti svoje informacijske potrebe in skrbno načrtovati razvoj informacijskega sistema s posebnim povdarkom na enotni in celoviti bazi podatkov. Le na ta način se lahko izogne kaosu, ki nujno sledi uporabi nove informacijske tehnologije na stari način.

3 PODATKOVNI PROTOTIPNI PRISTOP GRADNJE INFORMACIJSKIH SISTEMOV

Podatki so vir poslovnega sistema, enako pomembni, kot ostali viri, npr. denar ali kadri. Uspešnost poslovnega sistema je odvisna od ustreznega načrtovanja in krmiljenja podatkov in informacij, ki predstavljajo podlago računalniško zasnovanega informacijskega sistema.

Podatkovni prototipni pristop je ena od sodobnih metodologij gradnje informacijskih sistemov, ki ob gornjih ugotovitvah (ponovno) uvaja prototipni pristop na to področje. Gradnja informacijskega sistema se izvaja v dveh medsebojno povezanih in pogojenih fazah. Rezultat prve faze, faze načrtovanja, predstavlja izdelan in uveljavljen **strateški načrt gradnje informacijskega sistema**, ki v naslednji fazi zagotavlja hiter, učinkovit in informacijskim potrebam uporabnikov prilagodljiv **prototipni razvoj programskih rešitev**.

Načrtovanje informacijskega sistema mora biti vključeno v splošno načrtovanje poslovnega sistema. Izhaja iz ciljev poslovnega sistema in zgradi model sistema s podatkovnega vidika, ki je podlaga za gradnjo informacijskega sistema. Strateški načrt vzpostavi okvir, ki zagotavlja povezljivost posameznih sistemov po poslovnih področjih. Deli informacijskega sistema so grajeni ločeno, vendar morajo delovati kot celota.

Gradnja informacijskega sistema se izvaja po področjih in prioritetah, ki so opredeljene v strateškem načrtu. S sodobnim prototipnim pristopom zgrajeni informacijski sistem je dinamičen in prilagodljiv spremembam poslovnega sistema. Metodologija prototipnega pristopa zagotavlja postopno gradnjo baze podatkov in uporabniških rešitev s sprotnim preverjanjem ustreznosti zgrajenega sistema.

Pri načrtovanju in prototipni gradnji informacijskega sistema se uporabljajo sodobna metodološka znanja in informacijska orodja. Na ta način lahko zagotovimo izgradnjo informacijskih sistemov, ki so s stališča uporabnikov optimalni, medsebojno povezljivi ter se lahko dinamično prilagajajo novo nastalim potrebam poslovnega sistema.

Nove tehnološke možnosti, predvsem na področju mikro računalnikov, omogočajo gradnjo učinkovitih porazdeljenih (distribuiranih) sistemov, kjer se posamezni segmenti informacijskega sistema rešujejo s pomočjo lokalnih računalniških sistemov, ustrezna povezava pa ob pravilnem načrtovanju in vodenju zagotavlja delovanje informacijskega sistema kot integralne celote. Cilj porazdeljene obdelave je doseči napredek z decentralizacijo računalniških virov in centralizacijo standardov. Predvsem je potrebno zagotoviti

centralno kontrolo nad podatkovnimi strukturami, ki so skupnega pomena za poslovni sistem kot celoto in nad razvojem skupnih aplikativnih segmentov in mrežnih standardov. Premik v smeri porazdeljene obdelave mora povečati zanesljivost in učinkovitost in zmanjšati kompleksnost informacijskega sistema.

4 STRATEŠKI NAČRT INFORMACIJSKEGA SISTEMA

Cilj strateškega načrta informacijskega sistema je izgradnja izhodiščnega modela podatkov poslovnega sistema in opredelitev poslovnih področij, za katera se v naslednjih fazah gradnje informacijskega sistema razvijajo uporabniške rešitve po prototipnem pristopu.

Izhodiščni model podatkov prikazuje osnovne podatkovne razrede in njihove medsebojne povezave. Predstavlja podlago in okvir, ki zagotavlja povezljivost ločeno razvitih segmentov informacijskega sistema ter merilo za morebitni nakup izdelanih rešitev. Podatkovna osnova zagotavlja gradnjo informacijskega sistema, ki bo dinamično prilagodljiv spremembam v poslovnem sistemu.

Za dosego navedenih ciljev je potrebno izvesti naslednje aktivnosti:

Analiza poslovnega sistema

V okviru analize poslovnega sistema se izdela pregledni model poslovnega sistema, analizira cilje in probleme ter ugotovi kritične faktorje uspešnosti poslovanja.

Pregledni model poslovnega sistema prikazuje strukturo in delovanje poslovnega sistema. Opredeljuje poslovne funkcije, organizacijske enote, lokacije in skupine predmetov podatkov. Na ta način ugotovimo, kje se določene funkcije izvajajo, katere predmete podatkov uporabljajo, kako so funkcije in predmeti podatkov povezani s trenutno organizacijsko strukturo, itd.

Analiza ciljev in problemov je neposredno povezana s problematiko upravljanja poslovnega sistema. Analizo izvedemo po posameznih organizacijskih enotah. Na podlagi ciljev in problemov opredelimo informacijske potrebe.

Kritični faktorji uspešnosti so področja v poslovnem sistemu, od katerih je najbolj odvisna uspešnost in učinkovitost poslovanja. Na ta področja mora biti vodstvo poslovnega sistema posebej pozorno, tekoče stanje na njih pa mora biti merljivo in merjeno. Merjena stanja so zapisovana v podatke, informacijski sistem pa na osnovi njih izdeluje informacije, ki so potrebne za upravljanje poslovnega sistema.

Izdelava modela podatkov

Izhodiščni model podatkov informacijskega sistema je prikaz predmetov podatkov in njihovih medsebojnih povezav. Vsebuje predmete podatkov, ki bodo v fazi uvajanja informacijskega sistema vključeni v bazo podatkov. Ugotovljene predmete podatkov z medsebojnimi povezavami v obliki izhodiščnega modela podatkov uporabljamo v nadaljevanju strateškega načrtovanja.

Izhodiščni model podatkov postavimo v odnos z modelom poslovnega sistema, torej z organizacijsko strukturo, lokacijami in funkcijami. Odnos med modeloma neposredno opozarja na določene formalne nepravilnosti. Analiziramo tudi logično ustreznost povezav predmetov podatkov z organizacijsko strukturo, lokacijami in funkcijami.

Tako kreiran model razredov predmetov podatkov predstavlja okvir informacijskega sistema, ki bo grajen po posameznih segmentih. Ločeno razviti segmenti informacijskega sistema se bodo povezovali preko predmetov podatkov, zato je korektno izdelan izhodiščni model podatkov temelj za uspešen razvoj informacijskega sistema. Hkrati pa odraža naravno strukturo podatkov poslovnega sistema, ki je neodvisna od trenutnih postopkov, organizacijske strukture in informacijskih potreb. Zato zagotavlja gradnjo informacijskega sistema, ki bo stabilen in prilagodljiv spremembam v poslovnem sistemu.

Opredelitev področij informacijskega sistema

V tej aktivnosti se obravnavajo odnosi med predmeti podatkov in poslovnimi funkcijami. Odnose preuredimo tako, da dobimo naravno vezljive skupine poslovnih funkcij in predmetov podatkov.

S takšnim grupiranjem dokončno opredelimo osnovna poslovna področja, ki predstavljajo logično zaključene dele informacijskega sistema. Urejena matrika funkcije/podatki prikazuje tudi povezave med posameznimi segmenti informacijskega sistema. Te povezave v fazi gradnje zagotovimo z ustrezno bazo podatkov in pretokom podatkov znotraj celotnega informacijskega sistema.

Pri dokončni opredelitvi poslovnih področij natančno omejimo področja tako, da bodo dovolj majhna za hitro in uspešno uvedbo, ter hkrati dovolj velika, da ne bodo delila posameznih poslovnih funkcij.

Vpliv informacijske tehnologije na poslovanje

V analizi vpliva informacijske tehnologije opredelimo možnosti poslovnega sistema glede na pričakovan razvoj informacijske tehnologije. Opredelimo poslovne funkcije, na katere bo imela informatika največji vpliv. Pri tem je izhodišče pričakovan razvoj informacijske tehnologije, ki ga povežemo z definiranimi cilji poslovnega sistema.

V analizi tehnološkega okolja ocenimo računalniško opremo, ki je prisotna v poslovnem sistemu in potrebno nabavo nove opreme. Ocenimo tudi obstoječe uporabniške rešitve in rešitve, ki jih ponuja tržišče uporabniških rešitev. Potrebno tehnološko okolje povežemo s cilji povečanja uspešnosti poslovnega sistema.

Opredelitev prednostnih področij gradnje informacijskega sistema

Strateški načrt informacijskega sistema ugotavlja določena kritična in problematična poslovna področja. Projekte gradnje informacijskega sistema moramo obravnavati z vidika donosnosti naložbe, potrebnih vložkov ter tveganj.

Prednostna področja opredelimo glede na naslednje značilnosti:

- poslovna nujnost za avtomatizacijo področja ali za preoblikovanje obstoječega sistema,
- stroški in problemi vzdrževanja obstoječega sistema,
- nivo povezanosti področja s cilji poslovnega sistema,
- osebna pričakovanja vodstva,
- razpoložljivi viri za gradnjo informacijskega sistema.

5 RAZVOJ REŠITEV S PROTOTIPNIM PRISTOPOM

Gradnja informacijskega sistema se izvaja po področjih in prednostih, ki so opredeljene v strateškem načrtu. Metodologija prototipnega pristopa zagotavlja postopno gradnjo baze podatkov in uporabniških rešitev s sprotnim preverjanjem ustreznosti zgrajenega sistema. Tako nastali operativni informacijski sistem pa se lahko v primeru sprememb v poslovnem sistemu nadalje dograjuje s prototipnim pristopom.

Prototipni pristop pri gradnji informacijskega sistema izhaja iz ugotovitve, da je delujoč sistem potreben za razumevanje bistva problemov. Omogočen je s sodobnim razvojem informacijske tehnologije, z zmogljivo računalniško opremo in informacijskimi orodji. Uporabniške rešitve je mogoče hitro razvijati in prilagajati trenutnim in bodočim zahtevam poslovnega sistema in okolja.

Za dosego navedenih ciljev je potrebno izvesti naslednje aktivnosti:

- detaljna analiza stanja obravnavanega področja,
- zasnova oz. scenarij delovanja informacijskega sistema,
- zasnova baze podatkov,
- razvoj rešitve,
- uvedba rešitve,
- nadzor kakovosti v smislu doseganja informacijskih potreb in tehnoloških značilnosti.

Analiza stanja

Analiza stanja je namenjena podrobni proučitvi obstoječe organizacije, ustroja podatkov, poslovnih pravil in iz njih izhajajočih predmetov podatkov (entitet).

Za analizo stanja na področju odločanja je značilna potreba po vsebinskem opredeljevanju pravilnosti posameznih podatkov s stališča poslovnih pravil, izkušenj, znanja in uporabe. Vsebinsko nepravilni in za odločevalca neprimerno strukturirani podatki kaj lahko pripeljejo do neustreznih odločitev. Zato je potrebno podatke analizirati v smislu časovnih parametrov uporabe (zapadlost, veljavnost), analitične zmožnosti in uporabo (skupni, deljivi ali osebni podatki). V sodelovanju z uporabniki-odločevalci je

potrebno opredeliti njihove aktivnosti v smislu uporabe podatkov, analizirati probleme, s katerimi se srečujejo ob izvajanju in izvesti analizo kritičnih faktorjev uspešnosti. Slednji se nanašajo na področja v organizaciji, od katerih je najbolj odvisna uspešnost poslovanja.

Rezultat te aktivnosti predstavlja model obstoječega stanja, ki kot prva, osnovna varianta opisuje bodoči informacijski sistem. Model vsebuje:

- model aktivnosti znotraj poslovnih funkcij,
- model tokov podatkov v organizaciji,
- model podatkov na nivoju predmetov podatkov in njihovih povezav.

Zasnova informacijskega sistema

Izhodišče zasnove informacijskega sistema predstavljajo rezultati analize podatkov, kritične presoje obstoječega stanja in kritčni faktorji uspešnosti ter analize ovir za njihovo doseganje. Modificirajo se poslovna pravila, kar zahteva spremembe v modelu podatkov in modelu aktivnosti.

V zasnovi informacijskega sistema se opredelijo scenariji njegovega delovanja, ki prikazujejo način obravnavanja podatkov in so osnova za razvoj funcij informacijskega sistema.

Zasnova baze podatkov

Izhodišče zasnove baze podatkov, ki bo zagotavljala učinkovito uporabo podatkov različnih odločitvenih nivojev, predstavlja zasnovni model podatkovnih razredov in njihovih medsebojnih povezav. Rezultat je logični model baze podatkov, ki predstavlja prototipno bazo podatkov informacijskega sistema.

Na osnovi logičnega modela podatkov in izbranega krmilnega sistema baze podatkov se izdela fizična zasnova baze podatkov, ki predstvalja prototipno bazo podatkov. Baza, napolnjena s podatki, omogoča preizkus v realnem okolju.

Razvoj rešitve

Uporabniško programsko rešitev gradimo po prototipnem pristopu na osnovi uspešno zaključenih predhodnih aktivnosti. Osnovo rešitve predstavlja prototipna baza

podatkov in opredeljeni scenariji delovanja informacijskega sistema. Rešitev se razvija postopno, s sprotnim preverjanjem o ustreznosti zgrajenega sistema.

Uporabniki so aktivno vključeni v prototipni razvoj rešitve ter sproti potrjujejo njeno ustreznost in pravilnost. Na podlagi tako pridobljenih spoznanj in dodatnih zahtev uporabnikov se razvijejo vse popolnejše prototipne rešitve. Iterativni postopek se ponavlja do stanja, ko programska rešitev popolnoma zadovoljuje uporabnike v vseh njihovih informacijski potrebah.

Uvedba rešitev in nadzor kakovosti delovanja IS

Izdelane uporabniške programske rešitve je potrebno prilagoditi izbrani operativni računalniški opremi. Preveri se učinkovitost rešitev na dejanski količini podatkov ter izdela potrebne prilagoditve.

Opredeli se podatkovni vidik, ki zajema vprašanja celovitosti in varnosti podatkov:

- komu dovoljujemo dostop do podatkov,
- do katerih podatkov,
- kateri podatki so skupnega pomena za organizacijo, katere si uporabniki lahko medsebojno delijo in kateri so čisto osebni podatki,
- kako je organizirano skrbništvo baze podatkov in podatkov.

6 UPORABA INFORMACIJSKIH ORODIJ

Eden od ključnih pogojev gradnje sodobnih informacijskih sistemov, pri katerih zahtevamo postopnost gradnje s sprotnim preverjanjem ustreznosti aplikativnih rešitev, fleksibilnost in prilagodljivost spremembam informacijskih potreb, je uveljavitev prototipnega pristopa ob uporabi sodobnih informacijskih orodij.

Prototipni pristop je na področju gradnje informacijskih sistemov že dalj časa prisoten, vendar tehnološka raven uporabljenih informacijskih (bolje rečeno programskih) orodij ni omogočala njegove uspešne uveljavitve. Problem je izhajal prvenstveno iz pomanjkanja fleksibilnosti, počasnosti razvoja in neuporabnosti izdelanih prototipov kot delujočih rešitev.

V zadnjem času, s pojavom interaktivnih informacijskih orodij, so dani vsi tehnološki pogoji za polno uveljavitev prototipnega pristopa, ki omogoča sprotno in popolno vključitev uporabnikov v razvoj prototipov in gradnjo delujočih rešitev. Ta informacijska orodja v večini primerov slonijo na programskih jezikih četrte generacije, povezanih z relacijskimi bazami podatkov in katalogi podatkov.

Jeziki četrte generacije se razvijajo predvsem s ciljem dviga produktivnosti in uprabniške prijaznosti v primerjavi z uporabo postopkovnih jezikov prejšnje generacije (kot so FORTRAN, COBOL, PL/1...). Največkrat so nepostopkovni, kar pomeni, da z njimi "povemo računalniku" KAJ naj izvede namesto KAKO naj to izvede. Produktivnost razvoja rešitev se tako dvigne v povprečju vsaj za faktor 10. Največkrat zajemajo jezike za vzdrževanje baze podatkov, generatorje zaslonskih slik in poročil, poizvedovalne (query) jezike za pridobivanje ad-hoc podatkov iz baze podatkov. Nekateri so povezani z orodji za podporo odločanju, kot so ekstraktorji podatkov, preglednice, grafična orodja, knjižnice modelov odločanja in ostala matematično-statistična orodja.

Delovanje krmilnih sistemov baz podatkov oz. koncepta **relacijskih baz podatkov** sloni na medsebojno povezanih tabelah podatkov (datotekah), kjer se povezave med podatkovnimi elementi in datotekami dinamično prilagajajo potrebam uporabe. Takšno tehnologijo pa pogojuje uporaba prototipnega pristopa, kjer se opredelitve posameznih predmetov podatkov (entitet) in njihovih medsebojnih povezav često spreminjajo, dodajajo in brišejo, ne da je ob tem nujna ponovna zasnova in programiranje aplikativnih rešitev.

Katalogi podatkov predstavljajo informacijska orodja, v katerih se nahajajo vsi opisi in opredelitve podatkov, torej podatki o podatkih zajetih v bazi podatkov in ostalih podatkih informacijskega sistema. Posebno pri uporabi sodobnih podatkovno usmerjenih prototipnih metodologij gradnje informacijskih sistemov predstavljajo ključno razvojno orodje. Za uporabo v fazi načrtovanja so v nekaterih primerih povezani z orodji za podporo razvoja informacijskega sistema na logičnem nivoju (workbenches). V fazi gradnje in uporabe informacijskega sistema predstavljajo osnovo za vzpostavitev funcije skrbništva podatkov.

Izbira informacijskih orodij

Ob nekaterih predhodno ugotovljenih značilnostih sodobnih informacijskih orodij moramo ob izbiri oziroma odločitvi o uporabi upoštevati naslednje značilnosti:

- združljivost (kompatibilnost), ki izhaja predvsem iz usmeritve proizvajalca orodja k uveljavljanju standardizacije. Na tem področju je uveljavljen le standard poizvedovalnega jezika SQL, razen tega pa večina proizvajalcev, predvsem iz tržnih razlogov, teži k združljivosti z ustreznimi proizvodi firme IBM, kar nedvomno zagotavlja širšo uporabnost;

- prenosljivost, kjer je osnovni cilj uporaba informacijskega orodja, ki bo s stališča uporabnika enak ne glede na uporabljeni računalnik ali operacijski sistem. S stališča prototipnega pristopa h gradnji informacijskega sistema in uporabi v porazdeljenem (distribuiranem) konceptu je posebej pomembna neposredna prenosljivost rešitev med nivoji osebnih, srednjih in velikih računalniških sistemov, povezanih v omrežju;

- povezljivost s ciljem, da predstavljajo podatki, ki se nahajajo na različnih, medsebojno povezanih računalnikih, enotno porazdeljeno bazo podatkov. Pristop do kateregakoli podatka v takšnem sistemu mora biti s stališča uporabnika enoten in neopazen (transparenten), ne glede na operacijski sistem, komunikacijski protokol in krmilni sistem baze podatkov;

- produktivnost, tako v smeri razvoja aplikativnih rešitev po prototipnem pristopu, kot tudi v smeri učinkovitega pridobivanja podatkov za odločanje. Tudi tu je ključna enakost uporabe na celotnem spektru opreme od osebnega do velikega računalnika.

7 IZKUŠNJE PRI UPORABI PODATKOVNEGA PROTOTIPNEGA PRISTOPA

Pragmatičnim pristopom in klasičnim metodološkim pristopom h gradnji informacijskih sistemov, ki so bili in so še vedno prisotni v našem okolju, zaradi njihove neustreznosti sledijo poizkusi uvajanja sodobnejših, v svetu uveljavljenih pristopov. Medtem ko prvim lahko zamerimo nenačrtnost, parcialnost in nezmožnost povezovanja tako razvitih "aplikativnih otočkov", pri drugih, lahko bi jih imenovali tudi birokratskih, ki slonijo na klasičnem razvojnem ciklu gradnje informacijskega sistema, pa opažamo obsežnost, časovno nepredvidljivost in nefleksibilnost.

Klasičnim metodološkim pristopom, ob gornjih ugotovitvah, lahko še najbolj zamerimo nezmožnost ugotavljanja dejanskih informacijskih potreb uporabnikov. Praktične izkušnje kažejo, da žal tudi "dodatne" metodologije, ki naj bi predvsem dopolnile obstoje metodologije niso dosegle pričakovanih ciljev. Najdemo jih pod različnimi komercialnimi imeni kot so: ISAC, BSP, BICS, IA, izhajajo pa iz Langeforsove ugotovitve, da je potrebno za uspešno ugotavljanje informacijskih potreb predhodno izdelati model opazovanega poslovnega sistema.

Trenutno uporabljene metodologije gradnje informacijskih sistemov tudi pri nas slonijo na statičnem obravnavanju poslovnega sistema kot modela realnega sveta, kar pa za potrebe celovite in dolgoročne zasnove informacijskega sistema ni sprejemljivo.

Informacijske potrebe različnih nivojev in subjektov odločanja niso statične in enkrat za vselej predvidljive. Razen tega jih nikakor ni moč detaljirati in "zamrzniti" v začetnih fazah gradnje informacijskega sistema. V praksi se je pokazala moč podatkovnega prototipnega pristopa ravno v prilagodljivosti in prikladnosti bodočim uporabnikom informacijskega sistema.

Z uveljavitvijo novega pristopa smo povprečen vložek v izdelavo načrta razvoja, ki za razliko od klasičnega projekta operativno detajlira izvedbene korake, skrajšali na eno tretjino predhodno potrebnega časa in strokovnih človeških virov. Tudi na izvedbenem področju, ki izhaja iz takšnega pristopa, opažamo, ob uporabi sodobnih informacijskih orodij, podobne prihranke, da o kakovosti takšnih programskih proizvodov sploh ne govorimo.

9 UPORABLJENA LITERATURA

Appleton D.S.: Data-driven Prototyping, Datamation, november 1983

Kovačič A.: Postopki in izkušnje pri uporabi podatkovnega prototipnega pristopa h gradnji informacijskih sistemov, IEOS, Zveza ekonomistov Slovenije, Bled 1988

Kovačič A.: Tehnološko oblikovanje baz podatkov za potrebe odločanja v OZD, Raziskovalna naloga, VŠOD, Kranj 1986

Krstič D., A.Kovačič: Računalniška podpora oblikovanju informacij za poslovodni organ, Društvo ekonomistov Ljubljana, Portorož 1987

Lipp M.E.: Prototyping: A Data Driven Approach, Pergamon Infotech, marec 1985

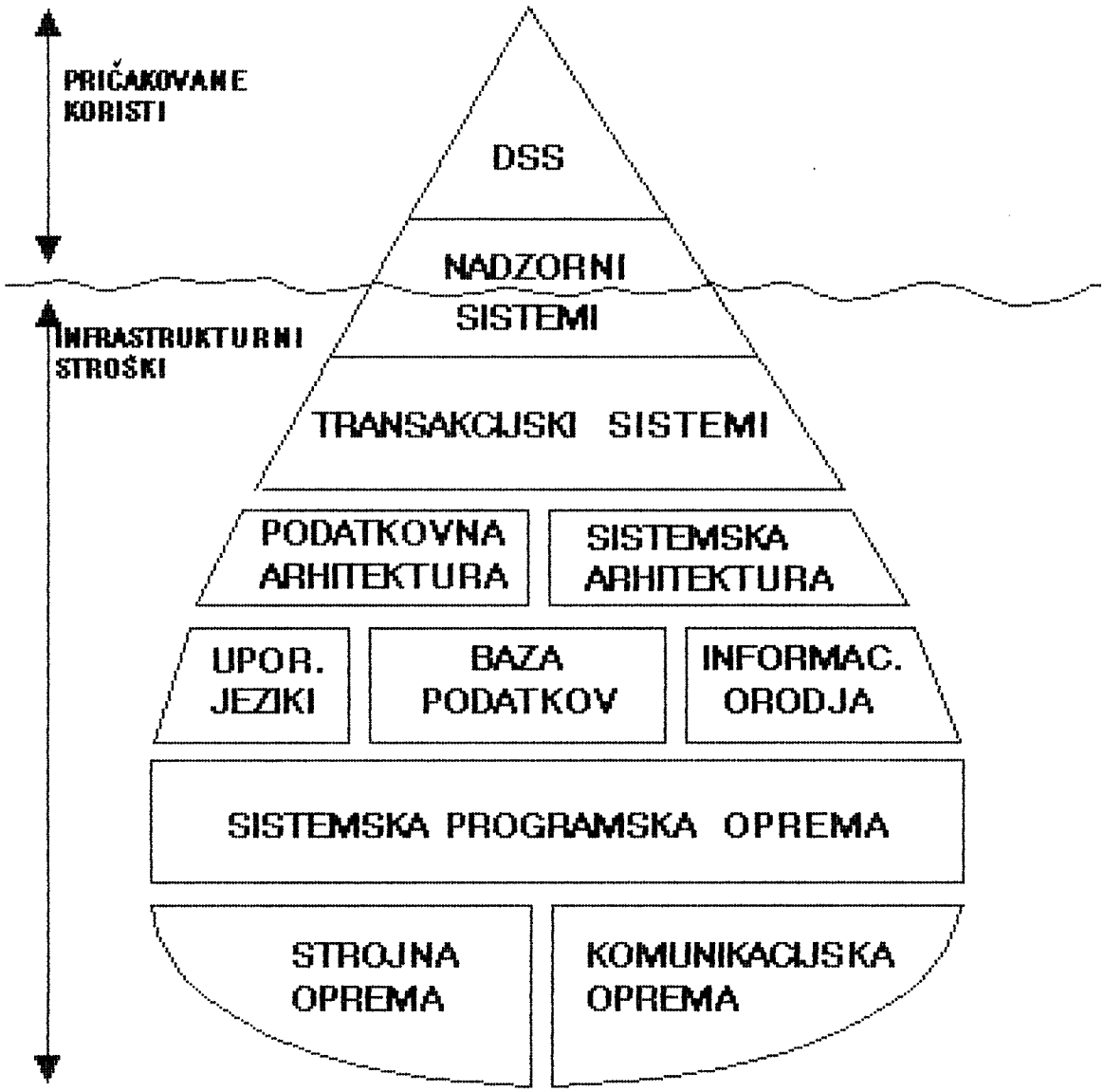
Navathe S.B., L. Kerschberg: Role of Data Dictionaries in Information Resource Management, Information & Management, januar 1986

Mc Fadden F.R., Y.F. Hoffer: Data Base Management, Benjamin-Cummings Publ., 1985

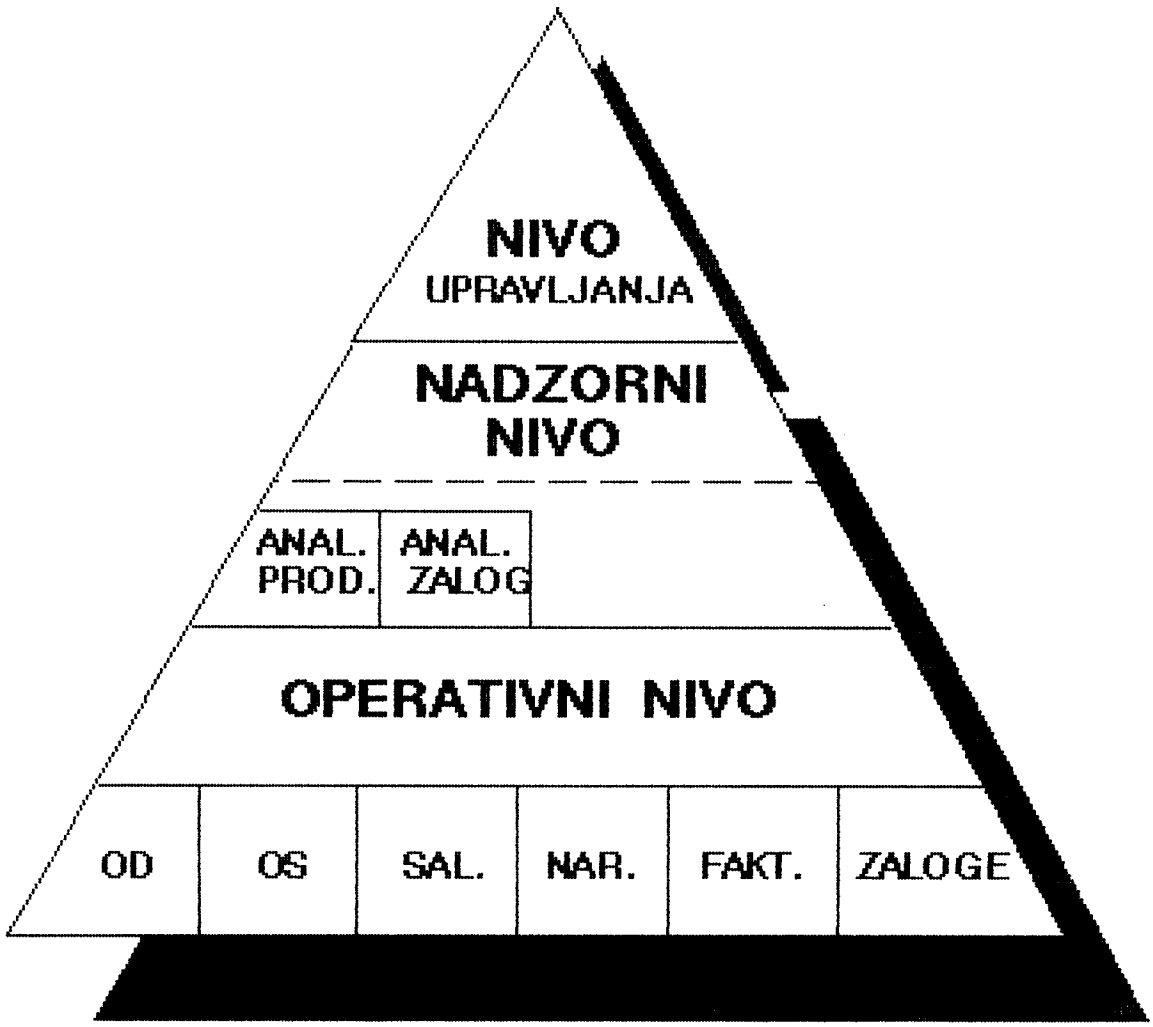
Yadav S.B.: Determining an Organisation's Information Reqirements, DATA BASE, pomlad 1983

----------.: Gradnja informacijski sistemov za podporo odločanja, Seminarsko gradivo, DO PRIS, Ljubljana, maj 1988

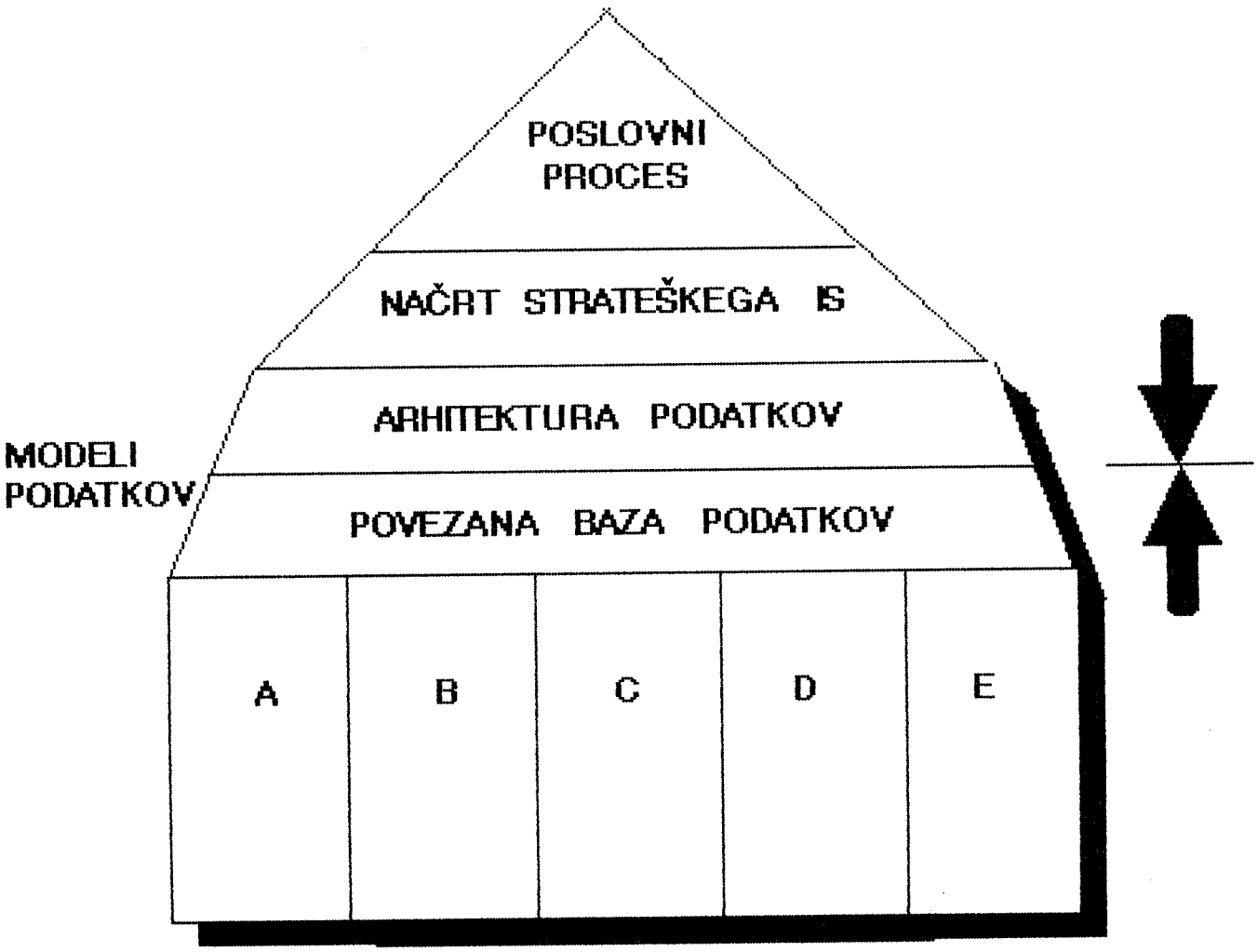
INFRASTUKTURNA LEDENA GORA



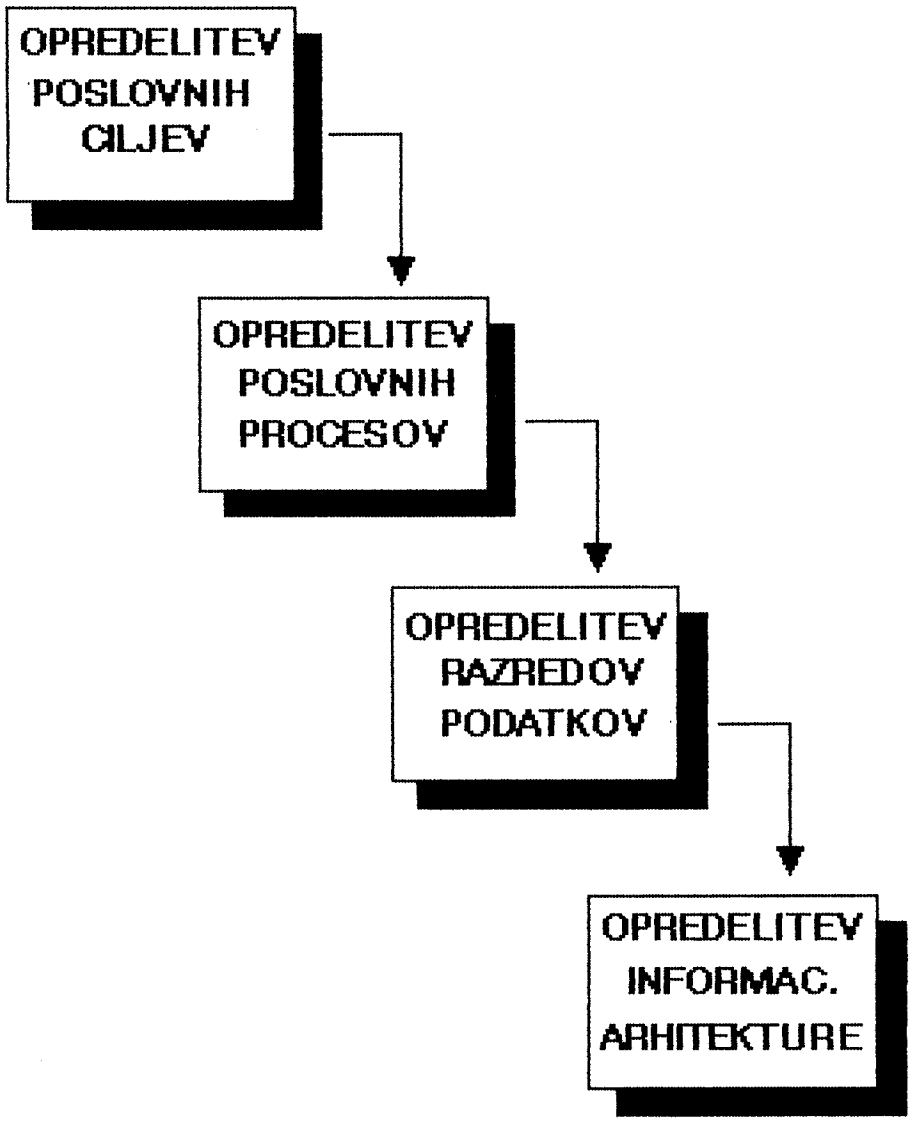
NIVOJI INFORMACIJSKEGA SISTEMA



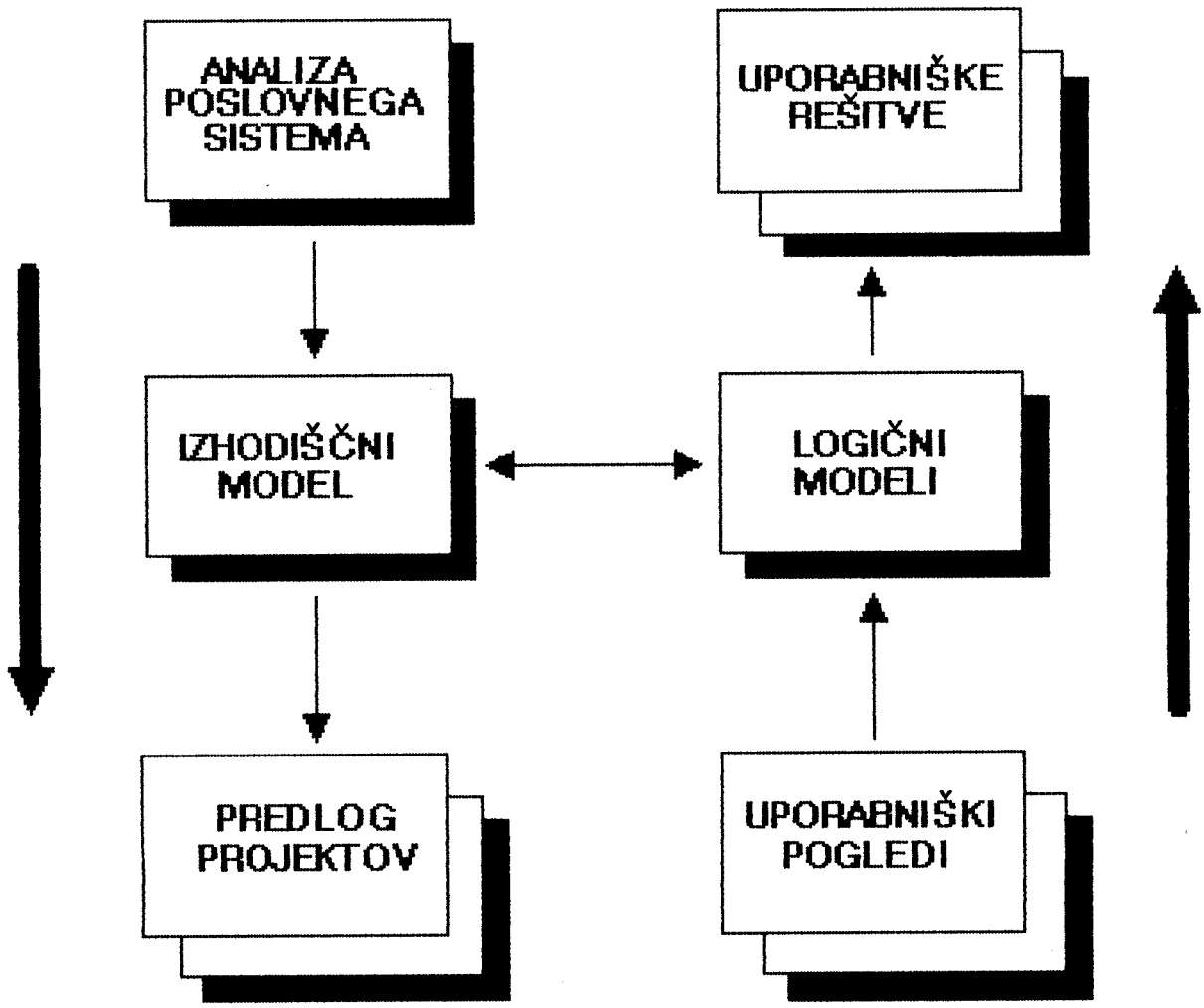
NAČRTOVANJE STRATEŠKIH IS



FAZE NAČRTOVANJA INFORMACIJSKEGA SISTEMA



GRADNJA INFORMACIJSKEGA SISTEMA



PRIS

MODEL PODATKOV

KUPCI

KUPEC	NAZIV	NASLOV	POSTA	KRAJ	TELEFON

ARTIKLI

ARTIKEL	NAZIV

CENIK

ARTIKEL	**ZACETNI DATUM VELJAV.**	KONCNI DATUM VELJAV.	STAND. CENA	MIN. CENA

NAROCILA

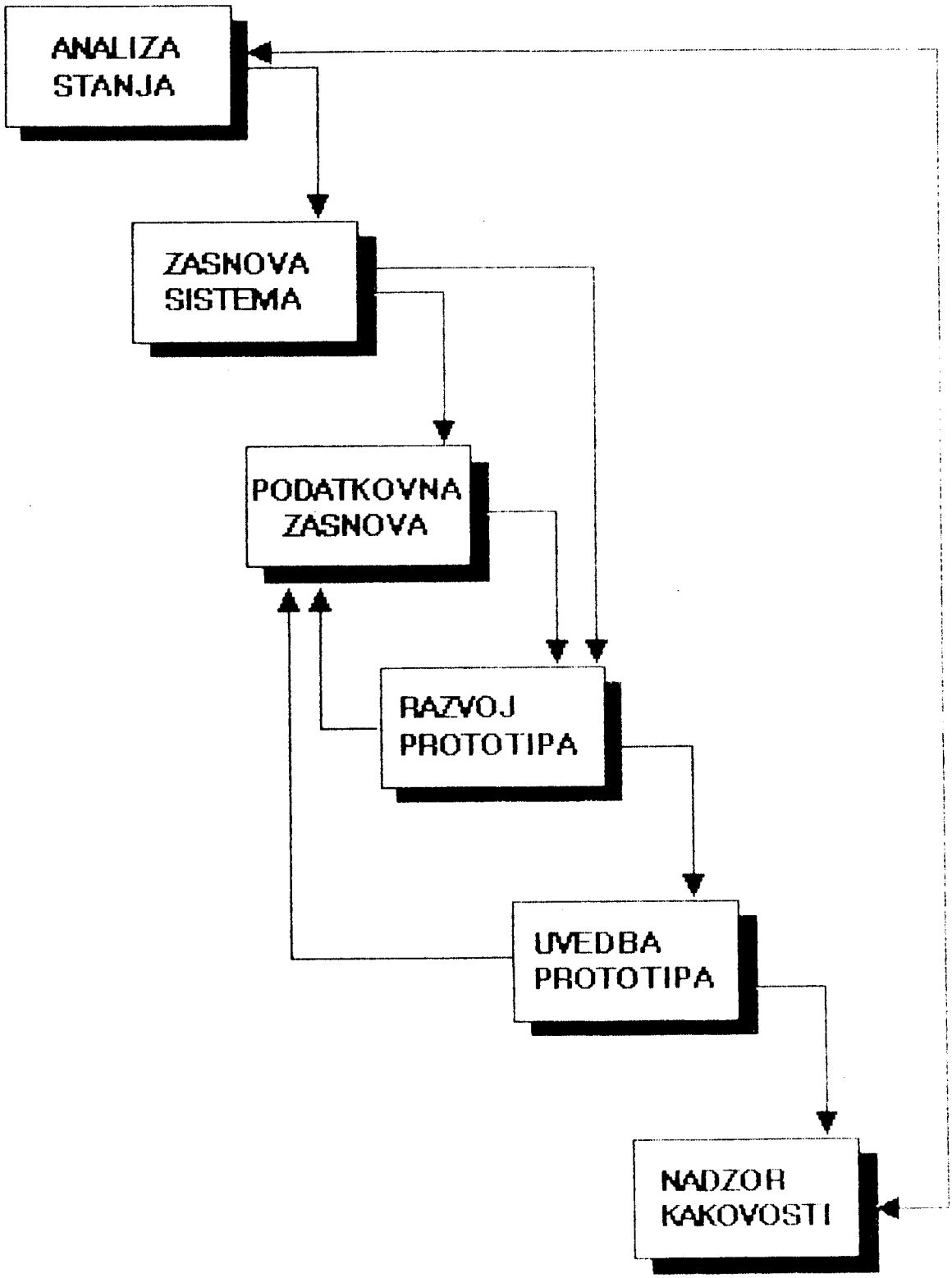
NAROCILO	KUPEC	DATUM NAROC.	ROK DOBAVE	VREDNOST	PRODA-JALEC	STATUS

POSTAVKE NAROCILA

NAROCILO	**POSTAVKA**	ARTIKEL	DEJANSKA CENA	KOLICINA	VREDNOST	DATUM DOBAVE

PRIS

FAZE RAZVOJA PROTOTIPA



PRIS

MODEL PODATKOV