INFORMACIJSKI SISTEMI

dr. Mirko Vintar

INFORMACIJSKI SISTEMI V UPRAVI

Informatika je postala glavna gonilna sila nadaljnjega razvoja poslovnih sistemov, kar velja tudi za poslovne sisteme v javnem sektorju, javni upravi. Lahko izhajamo iz nekaj specifičnosti, ki veljajo za javni sektor in zaradi katerih je pomen informatike, informacijskih sistemov in informacijske tehnologije v javni upravi še toliko večji. Pomembne točke razvoja javnega sektorja oz. državne uprave:

- ***Uprava kot storitveni in informacijski servis*** → uprava je usmerjena k čimbolj kakovostnim storitvam uporabnikom, postaja storitveni in informacijski servis.
- ***Izjemno velike baze podatkov in javni registri*** → uprava nudi vse več informacij občanom in organizacijam, v javnem sektorju se zbira velika količina informacij v bazah podatkov – potrebno je zagotoviti dostopnost informacij uporabnikom, ta dostopnost pa je v veliki meri povezana z informacijsko tehnologijo.
- ***Veliko število sočasnih uporabnikov*** → v javnem sektorju oz. v državni upravi je veliko število uporabnikov, ki imajo dostop do teh podatkov.
- ***Pojavljajo se podatki posebne narave*** → zaupni osebnimi podatki in podatki, ki so pomembni za varnost države.
- ***Informacijska tehnologija je postala v zadnjih letih temeljna tehnološka infrastruktura za delovanje sodobne javne uprave oz. gonilna sila nadaljnjega razvoja.***

POSLOVNI VIDIKI RAZVOJA IS

IS in poslovni sistem

DEFINICIJA → ***Informacijski sistem je skupek ljudi, postopkov in naprav, zasnovan za zbiranje, obdelavo, shranjevanje in distribucijo podatkov oz. informacij.***

Zavedati se moramo, da IS niso sami sebi namen, namenjeni so učinkovitemu delovanju in poslovanju. **IS ima podporno funkcijo. Podpirajo temeljno dejavnost vsake organizacije.** Ko razmišljamo, v kateri smeri mora iti razvoj IS, se moramo najprej vprašati, v katero smer se moramo kot organizacija razvija in kateri so razvojni cilji. Iz tega izhajajmo, kako z informacijsko tehnologijo in informacijskimi rešitvami podpreti načrtovani razvoj.

Informacijski sistem (poslovni IS) sestavljajo naslednji elementi:

- ljudje – pomemben akter,
- specializirani postopki, ki jih uporabljamo,
- množica naprav – računalniki.

V IS se odvijajo aktivnosti, ki so povezane z informacijami.
IS predstavljajo organizacijsko/tehnološko okolje za upravljanje z informacijami in informacijskimi tokovi obravnavanega poslovnega sistema.

***Poslovni sistem je organizirano okolje, v katerem se izvaja neka osnovna dejavnost*.** Vse kar je povezano z obdelavo informacij se odvija v okviru IS, zato vsaka organizacija za svoje delovanje oz. poslovanje potrebuje enega ali več različnih IS, ki vsebujejo vse podatke, ki jih potrebuje za svoje poslovanje. V praksi ima računovodstvo svoj IS, druge poslovne funkcije imajo svoje IS, ki pa so med seboj bolj ali manj povezani.

Vloga is v modernih organizacijah

Strateška vloga informatike in IS → igra pomembno vlogo tako v privatnem kot v javnem sektorju.
IS in konkurenčno sposobnost organizacije → IS so tisti, ki z inovativno uporabo IT in iskanjem novih rešitev, najbolj vplivajo na konkurenčno sposobnost organizacije. V javnem sektorju gre za storitveno dejavnost, zato tam zasledujemo kakovost storitev in učinkovitost delovanja.
Strateško načrtovanje IS → k načrtovanju tega področja je potrebno pristopati strateško in dolgoročno. Potrebno je sistematično načrtovanje tega področja, da bodo učinki takšni kot jih pričakujemo.

Kronološki razvoj uvajanja it v upravo

Prvi osebni računalnik je bil izdelan v začetku 80-ih let, in sicer ga je izdelal Bil Gates, podjetje IBM.
Pri nas v upravi pa so se prvi računalniki pojavili v začetku 70-ih let.

- **Obdobje avtomatizacije (1970 – 1990)** → šlo je avtomatizacijo preprostejših, ročnih opravil. Razvoj se je najprej pričel na področju davkov, registra prebivalstva, katastrov,… kjer je veliko ročnega dela. V tem času se je vse avtomatiziralo in sicer po posameznih aktovnostih.
- **Obdobje informatizacije (1990 – 2000)** → to obdobje je značilno za 90-ta leta. Med avtomat. in informat. pa obstaja pomembna razlika s stališča sprememb, ki jih prinašata v organizacijo poslovanja.
- **Obdobje razvoja e-uprave (2000 - ……)** → rast interneta v drugi polovici 90-ih let je upravi prineslo nove možnosti v razvoju. Veliko večino storitev bomo v prihodnosti lahko opravili preko interneta in elektronske pošte. Razvoj e-uprave pa bo še potekal v prihodnjih približno 5-ih letih.

Od avtomatizacije do e-uprave

V prvih 20-ih letih je bil ves razvoj usmerjen na **operativno raven**. Prvo obdobje (obdobje avtomat.) je temeljilo na velikih računalnikih za obdelavo podatkov. Razvoj osebnega računalnika pa je začel računalnik seliti na nova področja. Od leta 80 in naslednjih deset let niso dosti vplivali na delo, saj teh računalnikov ni bilo. Uporaba računalnikov je bila usmerjena predvsem na področje davkov, zemljiških katastrov, raznih registrov in na področje pisarniškega poslovanja.
Od leta 1990 pa se je začelo obdobje informat. in uporaba računalnikov se je začela širiti na **taktično raven** (strokovno delo referentov, finančno poslovanje organov, materialno poslovanje organov, spremljanje proračuna).
V letu 2000 pa preidemo v obdobje e-uprave. Razvijati so se začele neke nove možnosti, nove rešitve in stare ne bodo več uporabne. Vse rešitve, ki jih uprava uporablja, je potrebno razvijati na novo (uporaba interneta).

Od avtomatizacije v informatizacijo

IT se pri nas obravnava preveč ozko, kot da se še vedno uporablja obdobje avtomatizacije. Informatizacija pa prinaša v upravo bistvene spremembe. Še več sprememb pa bo prineslo obdobje e-uprave.

Razlika med procesom avtomatizacije in procesom informatizacije organizacij:

- *Način uvajanja* → Procesi uvajanja avtomatizacije so se vodili od spodaj navzgor, pobude za uvajanje avtomat. so prihajale od nižjega in srednjega do vrhovnega menedžmenta. Proces uvajanja informatizacije pa mora vedno potekati od zgoraj navzdol.
- *Vpliv na organizacijo* → avtomat. je imela razmeroma majhen vpliv na organizacijo, predvsem na operativni ravni, na višjih ravneh pa ga ni bilo. Informat. pa ima izredno velik vpliv na organizacijo in začenjajo se predvsem spremembe v sami organizacijski kulturi.
- *Potrebna tehnologija* → včasih smo imeli samostojne računalnike, danes pa imamo lokalne in globalne mreže, internet in intranet.
- *Iniciator sprememb* → v obdobju avtomat. je bil nižji in srednji management, v obdobju informat. pa vrhovni management.
- *Odgovornost za izvedbo* → včasih je bila na nižji in srednji ravni, danes pa se odgovornost prenaša na vrhovni managemetn.
- *Obseg sprememb v poslovnih procesih* → prej so ostali poslovni procesi praktično nespremenjeni, spremembe so bile majhne, predvsem v načinu izvajanja – ročna opravila se nadomeščajo z avtomatiziranimi, v obdobju informat. pa lahko govorimo o radikalni prenovi poslovnih procesov.
- *Baze podatkov* → včasih so bile parcialne po poslovnih funkcijah, danes pa so potrebne integrirane baze podatkov za celotno organizacijo.
- *Upravljanje informacijskih virov* → včasih je bilo decentralizirano po organizacijskih enotah ali poslovnih funkcijah, danes pa je centralizirano.
- *Vpliv na management* → prej vpliva na management skoraj ni bilo, danes pa je ta vpliv precej večji.
- *Vloga IT v organizaciji* → vpliv je bilo čutiti na operativni in tehnični ravni, danes pa IT dobiva strateško vlogo – vse vitalne funkcije neke organizacije so odvisne od uporabe IT.
- *Spremembe v organizacijski strukturi* → sprememb v strukturi običajno ni bilo, danes pa so te spremembe lahko zelo velike, kar pa je odvisno od narave organizacije.
- *Spremembe v normativni ureditvi* → te spremembe prej niso bile nujne, danes pa so koristne in so celo pogoj za uspeh projektov informatizacije.

Ključne točke procesa informatizacije organizacij

Informatizacija je kompleksnejši proces kot avtomatizacija, to je večdimenzionalen proces in sicer gre za:

- uvajanje informacijske tehnologije v vse faze zbiranja, obdelave, shranjevanja in posredovanja informacij
- prenovo poslovnih procesov na osnovi inovativne uporabe IT.
- preureditev informacijskih tokov ter njihovo prilagoditev možnostim IT.
- prilagoditev ali sprememba organizacijske strukture, v katero se uvaja sodobna tehnologija
- prilagoditev metod menedžmenta uporabi sodobnih informacijskih tokov

Proces informatizacije je prisoten v vseh organizacijah, v nekaterih je ta proces že dlje, drugje šele na začetku.

NAČRTOVANJE IN GRADNJA IS

METODOLOGIJE, METODE IN POSTOPKI PRI RAZVOJU IS

Metodologija je po definiciji skupek postopkov, tehnik in metod, ki jih uporabljamo pri reševanju nekega problema. Pod pojmom »metodologija gradnje IS« si vsaj v praksi največkrat predstavljamo organizacijsko-tehnično znanje, ki ga uporabljamo pri zasnovi in izdelavi računalniških rešitev.

Glavni elementi celovite metodologije → vsaka celovita metodologija bi morala opredeliti oz. vsebovati naslednje elemente:

- opredelitev ključnih razvojnih faz ter njihovega sosledja
- vsebinski zapis vsake faze z opredelitvijo ključnih aktivnosti
- navodila za izvedbo aktivnosti
- prikaz metod in tehnik za izvedbo posameznih aktivnosti
- opredelitev zahtevanih rezultatov posamezne faze
- opredelitev kriterijev za kritično ovrednotenje rezultatov posameznih faz
- navodila glede organizacijskih, kadrovskih ter tehničnih pogojev, ki so pomembni pri uporabi metodologije
- opredelitev področja uporabnosti.

Metodologije, ki bi v celoti izpolnjevala vse gornje elemente ter pokrivala vse razvojne faze, še ne poznamo.

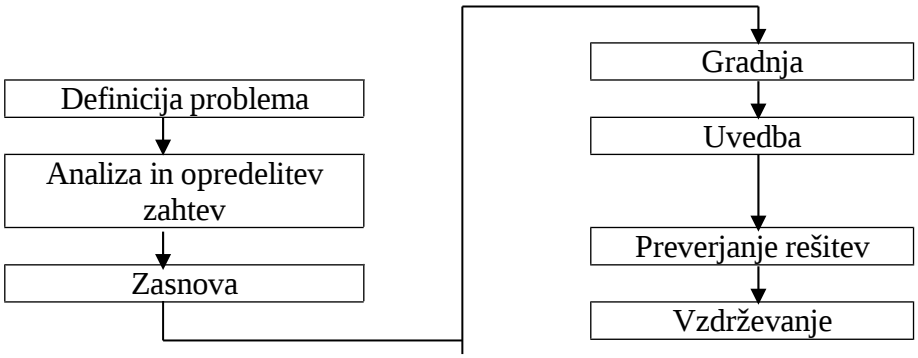
Ključni problemi razvoja IS

- »streljamo na tarčo v gibanju« → IS vedno razvijamo za konkretni poslovni sistem, okolje in ga je potrebno prilagoditi potrebam, da je smiseln. IS pa se danes še vedno razvijajo relativno dolgo, v tem času pa se razvija tudi okolje.
- nedorečenost metodologij in orodij → do cilja vodi različno veliko poti in uporabnik se mora sam odločiti, kako bo dosegel določen cilj.
- sodelovanje vodstva in uporabnikov → brez tesnega sodelovanja uporabnikov in vodstva ni uspešnega razvoja IS.
- predolgi razvojni cikli → živimo v izredno dinamičnem času, ko vemo, da v enem letu neka organizacija nastane, zaživi in tudi odmre. Po drugi strani pa kompleksnejše rešitve trajajo nekaj let, kar je s stališča menedžmenta nesprejemljivo, saj če želimo nekaj izboljšati, želimo rešitve čimprej. Problem je povezan z metodologijami in seveda z orodji, ki jih imamo na voljo. Glavni problem je, da predolgo traja od ideje do konkretne rešitve, ki jo lahko začnemo uporabljati.
- nepredvidljiva kakovost razvitih informacijskih rešitev → takrat, ko se takega problema lotimo, še vedno velikokrat ne vemo, kaj bomo na koncu v resnici dobili, ker so razvojni procesi zelo dolgi.
- visoki razvojni in vzdrževalni stroški – investicije za informacijsko infrastrukturo v celoti naraščajo iz leta v leto in postajajo v vseh organizacijah vedno večje breme. Enako velja tudi za vzdrževalne stroške. Najdražje je, če moramo vse razviti sami, najceneje pa je če kupimo na trgu, vendar pa je možno, da moramo potem prilagajati.

RAZVOJ IS

ŽIVLJENJSKI CIKEL IS

Vsak informacijski sistem gre v razvoju skozi določene karakteristične razvojne oz. življenjske faze. Govorimo o razvojnem oz. življenjskem ciklu IS. Življenjski cikel je nekoliko širši od razvojnega. Pod razvojnim ciklom razumemo faze od opredelitve problema do izvedbe. Življenjski cikel pa vsebuje še dve fazi: preverjanje rešitev in vzdrževanje (o tem govorili že pri informatiki).



Kako pa konkretno izvedemo posamezne razvojne faze, je odvisno od metodološkega pristopa. Nobena metodologija se ne drži tega življenjskega cikla tako, da bi si faze sledile v tem zaporedju, čeprav je to neko naravno zaporedje. Vsak metodološki pristop gre skozi svoje karakteristične razvojne faze. V sedanjem razvoju IS se je razvila množica različnih metodoloških pristopov.

3 karakteristični pristopi k načrtovanju in gradnji IS so:
- linearni pristop – najstarejši klasični pristop;
- prototipni pristop – se je razvil kasneje
- objektni pristop – zadnja leta govorimo o objektnem, čeprav se še vedno razvija. Posamezni elementi tega pristopa so že zreli za uporabo, kot celota pa se še uvaja

Linearni pristop

Linearni pristop sestavlja niz zaporednih faz, za katere je značilno, da se nobena ne more začeti, dokler ni v popolnosti dokončana predhodna faza. Po vsaki fazi se izdela poročilo, ki služi kot osnova za začetek naslednje faze.

Ta pristop se je začel razvijati koncem 70-tih let in se je ohranil vse do danes in je še danes temeljni pristop, ki ga uporabljamo pri razvoju IS.

Značilnosti linearnega pristopa:
- razvoj IS poteka po »kaskadnem principu«, skozi natančno določene faze, ki si sukcesivno sledijo
- vsaka razvojna faza je natančno definirana in dokumentirana
- je neodvisen od velikosti problema
- je neodvisen od uporabljenih orodij

Pri kaskadnem principu si predstavljamo tok vode več slapov. Ko je prva kaskada polna vode, se začne prelivati naprej itd – torej po fazah. To je slikovita primerjava tega, kar se dogaja pri razvoju IS. Pri linearnem pristopu se predpostavlja, da lahko razvoj IS razdelimo na nekaj

karakterističnih razvojnih faz, ki so natančno definirane in jih izvajamo po točno določenem zaporedju.

Dobre strani tega pristopa → faze so dobro dokumentirane, kajti osnova za vstop v vsako naslednji fazo je, da je predhodna natančno dokumentirana, kar nas sili v dobro dokumentiranje tega, kar smo naredili. Ta pristop je neodvisen od velikosti problema. Gre za sistematski pristop, ki ga uporabljamo pri preprostih rešitvah ali pa tudi pri najzapletenejših IS. Hkrati je tudi neodvisen od orodij, ki jih uporabljamo.

V praksi se je običajno izkazalo, da posameznih razvojnih faz ni mogoče tako natančno opredeliti niti niso mogoči tako čisti prehodi iz predhodne v naslednjo fazo. Pogosto se tudi izkaže, da predhodna faza ni bila opravljena dovolj temeljito. To zahteva pogosto vračanje nazaj v predhodne faze, spreminjanje že narejenega in dokumentiranega, kar povečuje stroške in podaljšuje čas za realizacijo projekta.
Vse to pa vodi k naslednjim **slabostim**:

- *predolgi razvojni cikli,*
- *visoki razvojni stroški,*
- *nepredvidljiva kakovost končnih izdelkov,*
- *sodelovanje uporabnikov je otežkočeno ali nemogoče.*

To so ključni problemi tega pristopa, ki so se pokazali zelo zgodaj, zato se je sredi 80-tih let razvil prototipni pristop.

Prototipni pristop

Pomeni močan korak v drugo smer in je nastal kot odgovor na slabosti linearnega pristopa.

Vloga prototipa → v industriji in tehničnih področjih se prototip uporablja že dolga leta. Noben resnejši izdelek ne nastane brez prototipa, na katerem se preizkusijo njegove lastnosti. Prototip se testira, ugotovi se slabosti in pomanjkljivosti, nato se načrti popravijo in šele potem se začne s serijsko proizvodnjo. Prototip pa se zavrže, služil je le v fazi razvoja nekega izdelka. Pri informacijskem pristopu ima prototip drugačno vlogo. Ne služi samo v fazi razvoja IS, ampak predstavlja tudi končno rešitev.

Pod prototipnim pristopom razumemo neke vrste **evolutivni pristop** → rešitev se razvija korak za korakom. V sodelovanju z uporabniki skušamo najprej razviti prototip sistema, ki ga potem dopolnjujemo in spreminjamo, dokler ne izpolnjuje vseh uporabnikovih želja. Prototip se v tem primeru postopoma razvije v končni proizvod.

Vloga uporabnikov je ključna, saj imajo ves čas možnost vplivati na razvoj in hkrati nosijo soodgovornost. Sodelovanje med razvojnim timom in uporabnikom je torej velika prednost. Na ta način je nevarnost, da bomo na koncu dobili nekaj drugega, kar smo načrtovali, zmanjšana na minimum.

Res pa je, da tu ne gre več za tako sistematičen pristop kot pri linearnem, kjer so faze natančno definirane. Tu gre bolj za improviziran pristop, pri katerem razvojne faze niso striktno določene.

S prototipnim pristopom se skuša doseči predvsem naslednje:

- skrajšati razvojni čas, ki je potreben da pridemo do prvih rezultatov projekta,
- znižati razvojne stroške,

- omogočiti kreativno sodelovanje uporabnikov skozi celotno razvojno obdobje projekta,
- zagotoviti, da se eventuelne napake in pomanjkljivosti pokažejo v zgodnjih fazah, ko jih je enostavno odpraviti.

Po drugi strani pa so **slabosti** prototipnega pristopa, da so navadno rešitve, ki jih dobimo slabo ali pa sploh nedokumentirane. Težko jih je vzdrževati in spreminjati. Zato se v praksi pokaže, da je ta prototipni pristop uporaben, kadar gre za razvoj manjših do srednje velikih IS, za razvoj kompleksnejših rešitev velikih IS ta pristop sam zase ne bo dovolj, zato ga lahko kombiniramo z linearnim pristopom.

Objektni pristop

Je bistveno drugačen od obeh prejšnjih in je nastal kot odgovor na slabosti obeh pristopov, zlasti linearnega. Ideja se je začela razvijati v začetku 90-ih let.

Njegov namen je skrajšati razvojne cikle, znižati razvojne stroške in izboljšati kakovost rešitev. To so osrednji cilji objektnega pristopa.

Sama ideja objektnega pristopa temelji na tem, da pri tem pristopu ne razvijamo IS povsem na novo in po nekem unikatnem načelu, pač pa v razvoj IS uvaja elemente standardizacije in tipizacije, s čimer bistveno skrajšamo čas procesa, znižamo stroške in lahko tudi povečamo kakovost izdelanih rešitev.

Objektni pristop temelji na **objektih**, ki vsebujejo podatkovne strukture in pripadajoče postopke na teh strukturah. Objekti so največkrat objekti, ki nastopajo v realnem svetu (produkti, kupci, občani,…) in ki jih lahko obravnavamo kot entitete. Entitete so objekt, subjekt ali pojmi, ki nastopijo v poslovnem sistemu in o katerih zbiramo podatke v okviru IS:
- pri kadrovskem IS so entitete: zaposlen, delovno mesto, izobrazba, …;
- pri študijskem IS: študent, predmet, študijska obveznost, predavatelj, ….;
- pri IS bolnice: pacient, pregled, zdravnik, diagnoza, ….

To so entitete, na katere se nanašajo podatki v okviru IS in te entitete v okviru objektnega pristopa so objekti, iz katerih se IS sestoji.

Prednosti objektnega pristopa:
- večkratna uporaba istih objektov, kar bo znatno skrajšalo čas za razvoj novih rešitev ter zmanjšalo stroške,
- zanesljivejše in kakovostnejše rešitve,
- ker so objekti povsem zaključene celote, se individualni objekti lahko spreminjajo, na da bi to kakorkoli zahtevalo spremembe v drugih objektih, kar zelo poenostavi vzdrževanje programskih rešitev.

ŽIVLJENSKI CIKEL RAZVOJA IS

Življenjski cikel razvoja IS se začne z nekim dokumentom – vzpostavitvenim dokumentom projekta (VDP). Najprej pripravimo ta dokument in predstavlja osnovo za razvoj.

Analiza in strateško načrtovanje sistema – študija upravičenosti (I. faza). V tej fazi vglavnem proučujemo lastnosti tega obravnavanega sistema in njegove informacijske potrebe in na splošno opredelimo želene značilnosti sistema. Ugotavljamo kaj potrebuje ta obravnavani sistem, da bo lahko učinkovito funkcioniral.

Nato pa se moramo odločiti ali bomo šli v lasten razvoj ali pa bomo na trgu poiskali rešitev, ki bo čim bližja tistemu kar potrebujemo. Lasten razvoj nas lahko pripelje do bistveno višjih stroškov in porabimo lahko zelo veliko časa; pri nakup, pa točno take rešitve, kot jo želimo ne najdemo in zato kar pa spet porabimo več časa in stroški so višji.

Če se odločimo za **nakup** sledi: prilagajanje rešitev, uvedba novega sistema, preverjanje rešitev in vzdrževanje. V vsaki od faz pa lahko ugotovimo, da nam nekaj manjka in se moramo vračati nazaj.

Če pa se odločimo za **lasten razvoj** sledijo še 3 faze razvoja IS:
- *logična zasnova*
- *fizična zasnova in razvoj*
- *uvedba IS* in nato še
- preverjanje rešitev in
- vzdrževanje.

V vsaki od faz pa se lahko zgodi, da se moramo vračati nazaj (v predhodne faze).

4-STOPENJSKI MODEL RAZVOJA IS

1. faza → ***analiza in strateško načrtovanje sistema – študija upravičenosti***
2. faza → ***logična zasnova*** (funkcionalno/vsebinski model IS)
3. faza → ***fizična zasnova in razvoj IS*** (gradnja IS)
4. faza → ***uvedba IS***

Analiza in strateško načrtovanje sistema ter študija upravičenosti
IS vedno razvijamo za neko konkretno poslovno okolje in za potrebe tega okolja. Vedno se moramo vprašati kam bi organizacija rada šla z razvojem →
- opredelimo strateške cilje organizacije
- naredimo strateški načrt informatizacije
- opredelimo funkcijo obravnavanega sistema
- naredimo analizo obstoječega stanja, ki zajema:
  - ✯ identifikacijo procesov in postopkov
  - ✯ analizo informacijskih tokov
  - ✯ grobo opredelitev informacijskih potreb
  - ✯ analizo obstoječih informacijskih rešitev in opreme
- sledi opredelitev strateških ciljev in
- ocena izvedljivosti oz. smiselnosti.

Logična zasnova IS

Ta faza je usmerjena v razvoj IS na vsebinski oz. funkcionalni ravni, v ospredju je funkcija in vsebina bodočega IS. Izhodišče so predvsem informacijske potrebe bodočih uporabnikov. Logična zasnova mora biti tehnološko in izvedbeno neodvisna

Rezultat te druge faze je **logični model IS**.
- Če gremo po poti tradicionalnega pristopa nastaneta dva modela → **postopkovni in podatkovni model**.
- Če gremo po poti objektivnega pristopa pa nastane en sam model → **objektivni model**.

Fizična zasnova in gradnja IS
Izhodišče za tretjo fazo je združitev vsebine in tehnologije in upoštevamo vse izvedbene ter tehnološke predpostavke. Izberemo tudi ustrezno strojno oz. programsko opremo in orodja.

Ob že izbranih orodjih, strojni in programski opremi, ki jo bomo uporabljali nastane **fizični model**, ki vključuje:
- razvoj baze podatkov,
- specifikacijo programskih modulov,
- zasnovo vhodov in izhodov podatkov (vhodne maske, izhodne maske/poročila).

Temu sledi **gradnja IS**, kar pomeni:
- programiranje,
- testiranje programov (formalno in logično testiranje),
- testiranje celotnega sistema.

in uvedba novega IS.

Rezultat zadnje, četrte faze pa je **izvedbeni model**.

LOGIČNA ZASNOVA IS

Izhodišča:
- jasno morajo biti opredeljeni strateški cilji
- informacijske potrebe morajo biti v grebem opredeljene (katere informacije mora IS zagotavljati, da bo delo učinkovito)
- opredeljena morajo biti poslovna pravila
- izbrati moramo metodološki pristop (po kateri poti bomo izdelovali IS).

MODELIRANJE INFORMACIJSKIH SISTEMOV

Pomembni pogledi na poslovni sistem z vidika razvoja IS

- ***organizacijski pogled*** → pokaže nam formalno organiziranost, organizacijsko strukturo; pokaže nam določene omejitve, ki so prisotne pri izvajanju postopkov
- ***normativni pogled*** → pomemben je v upravi, v podjetjih pa ima manjšo vlogo; pove postopke, dejanja, ki jih izvajamo v postopku upravnega delovanja – ZUP
- ***funkcijski pogled*** → delitev in struktura poslovnih funkcij, procesov in postopkov;
- ***procesni pogled*** → je podroben pogled v samo izvajanje postopkov; algoritmi, potek dela
- ***podatkovni pogled*** → pokaže vsebino podatkov in njihovo strukturo
- ***kontrolni pogled*** → integracija procesnega, organizacijskega in podatkovnega pogleda

Procesni in podatkovni pogled sta ključna.

Predstavitev pogledov na poslovni sistem

Za predstavitev pogledov na poslovni sistem uporabljamo različne metode in tehnike:

- organizacijski pogled predstavimo z **organigramom** – strukturni graf, ki nam pokaže hierarhično strukturo organizacije
- strukturni pogled predstavimo s **strukturnim grafom**
- procesni pogled predstavimo z diagrami poteka, ki so pomembni za:
  - **EPC** (event-proces-chain)
  - **DTP** (diagram toka podatkov) – vpogled v izvajanje procesov
- podatkovni pogled predstavimo z **E-R diagrami**, slovarji
- kontrolni pogled predstavimo z **extended EPC**

Posebno pomembni pogledi za razvoj IS

Procesni pogled je pomemben iz vidika ***razvoja programov*** in daje vpogled v:

- *poslovne procese*
- *postopke*
- *pravila in pogoje izvajanja postopkov*
- *podatke potrebne za izvajanje.*

Podatkovni pogled pa je pomemben iz vidika ***razvoja podatkovne baze*** in daje vpogled v:

- *informacijske potrebe*
- *entitete, atribute in povezave.*

Preslikava stvarnosti v model

IS predstavlja preslikavo realnega sveta oz. nekega sistema. Sam proces preslikave je dvofazen. Model, ki nastane, pa v resnici ni nikoli direktna preslikava stvarnosti, ampak je v bistvu predstava preslikave o stvarnosti v očeh opazovalca. Pri preslikavi pride torej do določenega popačenja in je opis stvarnosti lahko različen.

Preslikava predstave o stvarnosti v model

Model je vedno rezultat preslikave predstave o stvarnosti. Na to, kakšen bo ta model, lahko vplivata dva faktorja:

- ***»pogled na svet«***, ki ga ima opazovalec oz. razumevanje problema, ki je od človeka do človeka različno (vsakdo predlaga drugačno rešitev);
- drugi ključni dejavnik pa je ***instrumentarij***. Čim boljša orodja imamo na voljo, boljši bo model.

Na prvi dejavnik nimamo veliko vpliva, imamo ga pa na instrumentarij. Od tega je dejansko zelo odvisno, kako kakovosten bo model.

Instrumentarij, ki ga uporabljamo, so različni koncepti modeliranja IS (v knjigi razširjen seznam konceptov modeliranja IS: objekt, grupa, entiteta, atribut, vrednost, povezava, dogodek, začetni pogoj, končni pogoj, postopek, operacija, transakcija, aktivnost, zunanja entiteta, zbirka podatkov, informacijski tok, sporočilo, element).

Seznam konceptov modeliranja IS

1. entiteta 2. atribut 3. povezava 4. vrednost	vezani na modeliranje podatkov
5. postopek 6. zunanja entiteta 7. zbirka podatkov 8. informacijski tok 9. dogodek 10. sporočilo	vezani na modeliranje postopkov

S temi koncepti predstavimo na eni strani lastnosti podatkov, na drugi pa lastnosti postopkov. Prvi štirje koncepti so vezani na modeliranje podatkov, ostali pa na modeliranje postopkov. To je temeljni instrumentarij, ki ga uporabljamo. Razvoj IS naj bi rezultiral model logičnega IS, ki ga predstavimo z dvema modeloma: modelom podatkov in modelom postopkov. Ko razvijamo IS, v praksi dejansko oba modela nastajata več ali manj paralelno. Analiziramo poslovanje, analiziramo poslovni sistem, ugotavljamo kateri procesi in postopki se v okviru tega izvajajo.

RAZVOJ MODELA PROCESOV

Opredelitev poslovnega procesa

Poslovni proces je strukturirana množica aktivnostih, katerih rezultat je nek proizvod ali storitev s tržno vrednostjo. Zajema vhode in izhode, ki predstavljajo neko dodano vrednost za uporabnike. Praviloma se sestoji iz več postopkov in posega na več funkcijskih področij.

V upravi je tržna vrednost nekoliko odveč, ker uprava ni zasnovana v tej smeri. Razvoj uprave pa gre prav tako v to smer, da uprava postopoma prevzema gospodarstvo. Znano je, da je izvajanje poslovnega procesa, skozi katerega pridemo do določenega proizvoda, povezano s stroški. Ta logika je v podjetjih že dolgo znana in razvita, v upravi pa bomo rabili več časa, da se bomo navadili na to isto logiko. Delo v upravi razvijati skozi poslovne procese, neke aktivnosti, ki povzročajo stroške in da so stroški ravno tako pomembni, kot v gospodarstvu. Rezultat teh aktivnosti je prav tako proizvod oz. storitev. Zato je ta opredelitev poslovnega procesa zelo pomembna.

V upravi pojma proces ne poznamo, poznamo pa pojem upravni postopek. V bistvu je upravni postopek ravno tako proces, ki je strukturiran in ima jasne storitve (odločba).
Proces zajema vhode in izhode, ki predstavljajo dodano vrednost za uporabnike – če izhod ne predstavlja dodane vrednosti za uporabnike, ga je treba opustiti.

Struktura poslovnega procesa

Proces je širši pojem in praviloma so sestavljeni iz več postopkov. Med postopki in procesi je hierarhija. Če analiziramo poslovni proces, potem pridemo do večnivojske strukture. Proces se deli na:

- *postopke*,
- *aktivnosti in*
- *naloge.*

Problem pa je, kako priti do te strukture? Procesi so slabo ali pa sploh niso dokumentirani in v upravi ni nekih organizacijskih predpisov za njihovo izvajanje.

Funkcijska dekompozicija

Da lahko pridemo do strukture procesa, se poslužujemo metode, ki ji pravimo ***funkcijska dekompozicija***. To je sistematična analiza in razstavljanje poslovnih procesov na njihove poslovne dele, dokler ne pridemo do elementarnih sklopov. Praviloma jo izvajamo »od zgoraj navzdol«. Tem elementarnim sklopom pravimo elementarni postopki, ki se izvajajo v okviru nekega procesa.

Elementarni postopek je skupek medsebojno povezanih aktivnosti, ki se jih ne da ali ki jih ni smiselno razstaviti na manjše dele oz. sklope. Cilje je razstaviti na elementarne bloke. Pripelje nas do hierarhične strukture. Strukturo samega procesa je potrebno natančno predstaviti in ena od temeljnih tehnik, ki jo uporabljamo za predstavitev, je t.i. strukturni graf.

Funkcijska dekompozicija nas torej pripelje do elementarnih postopkov, ti pa so lahko večji, sestavljeni ali pa manjši oz. elementarni. Skoraj na vsakem področju bomo prišli do postopkov, ki so navadno različni. Nas zanima analiza postopkov z vidika razvoja IS, ki ga potrebujemo za izvajanje nekega poslovnega procesa. V tej strukturi nas zanimajo predvsem informacijski

postopki znotraj nekega poslovnega procesa, s katerimi se zbirajo, obdelujejo, spreminjajo in distribuirajo podatki, ki so povezani s tem poslovnim procesom. Te postopke je potrebno identificirati in podrobno opredeliti.

Opredelitev postopkov

Poudarek je na informacijskih postopkih, pri katerih se izvajajo operacije na podatkih. Bodisi se podatki zbirajo, spreminjajo, obdelujejo, distribuirajo, izmenjujejo, ...

Postopek predstavlja smiselno zaključeno množico operacij na podatkih, s katerimi se IS pripelje iz enega v drugo stanje. Opredeljen je z algoritmom in vhodno-izhodnimi podatki.

Da je nek postopek natančno opredeljen, moramo opredeliti vrstni red. Postopek mora biti opredeljen z algoritmom, s pravili za njegovo izvajanje. Če hočemo izvajati postopek, moramo vedeti pravila njegovega izvajanja.

Opredelitev osnovnih konceptov modeliranja postopkov

Algoritem je osrednji element definicije takega postopka, vendar pa sam algoritem ni dovolj. Za to, da bi postopke lahko podprli z informacijsko tehnologijo, je potrebno opredeliti še celo vrsto drugih stvari. Eden od konceptov, ki je pomemben pri opredeljevanju postopkov je dogodek.

Dogodek je koncept, s katerim skušamo opredeliti, kaj sproži izvajanje nekega postopka v praksi (npr. prispetje čeka, občan na okencu - postopek o izdaji gradbenega dovoljenja, izdaja potrdila o vpisu – študent v referat; .....). ***Dogodek je neke vrste impulz, do katerega pride bodisi zaradi sprememb ali posegov iz okolice IS ali zaradi notranjega stanja IS. Povzroči izvedbo enega ali več postopkov, s katerimi se spremeni stanje sistema***. Nekateri avtorji delijo te dogodke celo na začetne in končne dogodke. Za vsak postopek lahko opredelimo začetni dogodek, ki povzroči izvajanje postopka in dogodek oz. stanje, ki nastane po tem, ko je postopek izveden.

Za postopke je zelo pomembno opredeliti t.i. začetne in končne pogoje. **Začetni pogoj** nam ***opredeljuje vse pogoje, ki morajo biti izpolnjeni, da je zagotovljen pravilen potek izvajanja postopka.***
Končni pogoj ***opredeli vse pogoje, ki morajo biti izpolnjeni po izvedbi postopka, da se lahko šteje, da je bil postopek pravilno izvršen in normalno zaključen, ter da je zagotovljena integriteta IS*** (pri upravnem postopku je to zelo natančno določeno – začetni pogoj je popolna vloga – končni pogoj je pravnomočna odločba, ki potrjuje, da je bil postopek v celoti in pravilno izveden). Pri informatizaciji postopkov so ti pogoji zelo pomembni. Ne moremo razviti dobre informacijske rešitve, če teh pogojev ne poznamo.

Naslednji pomemben koncept je sporočilo. **Sporočilo *je zaključena celota informacij, ki vstopa v postopek ali izstopa iz njega. Je predmet obdelave v postopku in je potrebno za njegovo izvedbo, ali pa je rezultat njegove izvedbe***. Sporočilo je koncept, ki opredeljuje, kako postopki komunicirajo med seboj. Ta se delijo na vhodna in izhodna. Vhodna vstopajo v postopek in so potrebna za izvedbo postopka, izhodna sporočila pa so tista, ki izstopajo iz postopka in nastanejo kot rezultat izvedenega postopka.

Pregled metod in tehnik modeliranja postopkov

Strukturni graf
Strukturni graf je instrument za predstavitev funkcijskega pogleda. Je hierarhični graf, ki predstavlja celoto postopkov obravnavanega poslovnega procesa. Tak graf ima običajno več nivojev. Sestoji se iz vozlišč in povezav med njimi. Vozlišča predstavljajo bodisi elementarni ali sestavljeni postopek. Na splošno velja pravilo: vsi postopki, ki nastopajo na koncu vej, so elementarni. Tisti, ki se razstavlja naprej je sestavljeni. Vsi postopki znotraj morajo biti oštevilčeni. Pomembno je, da ima vsak postopek enolično oznako in da število narašča s številom nivojev. Iz grafa razberemo katere aktivnosti nastopajo v okviru procesa in kakšna je hierarhija med njimi.

Strukturni graf je samo ena od tehnik, ki jo uporabljamo pri strukturi poslovnih procesov in sama zase ni dovolj. Še veliko bolj pomembno je poznati informacijske tokove, ki nastopajo bodisi znotraj procesa, ali med procesom in njegovo okolico. Zato se poslužujemo drugih tehnik in ena najbolj razširjenih tehnik, s katero predstavimo podatkovne oz. informacijske tokove v okviru obravnavanega poslovnega procesa je diagram toka podatkov.

Diagram toka podatkov (DTP)
Diagram toka podatkov (data flow diagram) omogoča opredelitev vseh informacijskih tokov, ki nastopajo v okviru obravnavanega procesa ter med obravnavanim procesom in njegovo okolico.

Diagram toka podatkov je že dolga leta stalna in uveljavljena tehnika za predstavitev podatkovnih tokov in zbirk podatkov, ki nam nastopajo v okviru obravnavanega poslovnega procesa. Temelji na štirih konceptih oz. elementih:

- postopek
- zunanja entiteta
- zbirka podatkov
- tok podatkov
    - zunanja entiteta – postopek
    - postopek – zbirka podatkov

Temeljni element takega diagrama so **postopki** – odvisno od ravni obravnave – lahko pa tudi procesi, aktivnosti, ...

Drug element je **zunanja entiteta**. Pod zunanjimi entitetami razumemo vse entitete, ki nastopajo v okolici obravnavanega sistema in s katerimi ta obravnavani sistem komunicira (izmenjuje podatke, izmenjuje dokumente). Med zunanjimi entitetami in postopkom potekajo informacijski tokovi – v eni, v drugi ali v obeh smereh. Ni nujno, da je zunanja entiteta fizična oseba, lahko je tudi pravna oseba oz. vse kar obstaja v realnem svetu. To je zunaj sistema, s čimer sistem komunicira in izmenjuje podatke.
Zbirka podatkov – s tem konceptom opredelimo vse zbirke podatkov, ne glede na njihovo obliko, vsebino ali medij. Vedeti pa moramo, da podatki nastopajo med našim postopkom in na

drugi strani z našo okolico, zato tokove delimo na eksterne in interne. Na relaciji zunanja entiteta – postopek (med okolico sistema in sistemom) in znotraj samega postopka – zbirka podatkov.

Diagram poteka
Diagram poteka je ena najstarejših tehnik prikazovanja, kako se nek postopek izvaja. Daje nam vpogled v pravila izvajanja nekega postopka ali procesa.

Diagram poteka nam ne pove kdo izvaja posamezne aktivnosti, zato ga je možno tudi razširiti → razširjen diagram poteka.

EPC diagrami
So podobni diagramu poteka in kažejo strogo zaporedje izvajanja aktivnosti, v njih pa se pojavijo še dogodki, ki sprožijo ali omogočijo izvajanje procesa.

Kontrolni pogled

Kontrolni pogled združuje procesni, podatkovni in organizacijski pogled. Prednost kontrolnega pogleda je, da veže aktivnosti ter organizacijske in informacijske vire za njihovo izvedbo. Vezanje na aktivnosti in dogodke. Za ta pogled uporabljamo razširjeni EPC diagram.

MODELIRANJE PODATKOV

Modeliranje podatkov in podatkovni modeli so se razvijali vzporedno z razvojem informacijske tehnologije. Najprej so se razvili preprosti modeli za predstavitev fizičnih struktur podatkov na spominskih medijih. Z razvojem podatkovnih baz pa je pomembnost modeliranja močno narasla. Že v 70-ih letih so spoznali, da je pri modeliranju podatkov najpomembnejše čim bolj verno predstaviti njihov pomen, njihove povezave. Vse skupaj pa je vodilo v razvoj novih konceptov in modelov, ki omogočajo boljšo predstavitev pomena ter vsebine – semantike podatkov.

Podatkovni model

Podatek je opis, zapis nekega pojava ali dejstva in je lahko predstavljen v številčni, tekstovni ali grafični obliki. Podatki, s katerimi opisujemo pojave, dejstva, se nanašajo na neke objekte oz. entitete, ki nastopajo v okviru obravnavanega dela stvarnosti in so pomembni z vidika obravnave.

Rezultat modeliranja podatkov je **podatkovni model.** ***Podatkovni model je zbirka konceptov, s katerimi skušamo izraziti statične in dinamične lastnosti podatkov v okviru obravnavanega sistema (IS).***

Pod *statičnimi lastnostmi* razumemo vsebino in strukturo podatkov, ki se nanašajo na neko entiteto oz. objekt (vpisna št. študenta, priimek in ime, rojstni datum, spol, ….).
Pod *dinamičnimi lastnostmi* pa razumemo pravila spreminjanja teh podatkov (sprememba naslova, št. opravljenih izpitov, ….) Za podatke, ki se lahko spreminjajo, veljajo določena pravila, kdaj se lahko spremenijo in pod katerimi pogoji.

Podatkovni modeli so v informatiki zelo pomembni, še posebej na področju poslovne informatike, kajti IS na tem področju (gospodarstvo, uprava) običajno upravljajo z veliko količino podatkov, z velikimi bazami podatkov, katerih upravljanje je zelo zahtevno. Zato morajo biti zelo skrbno načrtovane in zgrajene.

Osnova za razvoj baze podatkov je podatkovni model, ki opredeljuje lastnosti podatkov. Razvoj podatkovnih modelov se je začel zelo zgodaj, v bistvu vzporedno z razvojem drugih programskih rešitev in vzporedno z razvojem strojne opreme. Razvoj strojne opreme je prinašal vedno večje in zmogljivejše računalnike, zmogljivejše spominske medije. S tem so nastajale možnosti za razvoj vedno večjih, obsežnejših in kompleksnejših sistemov in v teh sistemih nam nastajajo vse večje in kompleksnejše baze podatkov. In za razvoj te baze podatkov potrebujemo čimbolj učinkovita in čim boljša orodja – podatkovne modele.

Pri modeliranju podatkov na logični in konceptualni ravni je najpomembnejše, da je uporabljeni model čim bolj preprost, da ga razumejo tudi netehnično izobraženi uporabniki, hkrati pa mora omogočati verno predstavitev semantike podatkov. Najpogosteje uporabljen model je model entiteta-povezava oz. **E-R model**.

Z modeliranjem podatkov skušamo čim bolj verno predstaviti neki izsek realnega sveta ter njegove lastnosti. Pri tem se nam kot osnovni koncepti pojavljajo:

- *entiteta,*
- *atribut,*
- *povezava,*
- *vrednost.*

Entiteta je nek objekt, subjekt ali pojem, ki obstaja v realnem svetu in je pomembna z vidika načrtovanega IS. Entitete so lahko fizične ali abstraktne narave.
Pomen v informatiki → Entiteta je nek objekt, subjekt ali pojem, ki je pomembna za naš poslovni sistem, v katerem se zbirajo podatki.
Podatki v okviru konkretnega sistema, ki ga modeliramo, se nanašajo na entitete.

Atribut je opisna lastnost nekega tipa entitete, ki jo lahko pripišemo celotni množici primerkov danega tipa. Entitete opisuje končna množica atributov

Povezava je neka zveza med dvema ali več tipi entitet, ki je pomembna z vidika obravnavanega IS. Povezave me dvema tipoma entitet imenujemo binarne, ki pa so najpogostejše.

Splošni koncepti abstrakcije pri modeliranju podatkov

Abstrakcija je način razmišljanja oz. reševanja problema, pri katerem zavestno zanemarimo podrobnosti in se osredotočimo na splošne, skupne lastnosti pojavov, objektov ali pojmov, odvisno od narave problema, ki ga rešujemo. Tak pristop je še posebno pomemben na področju IS, saj gre za sisteme, v katerem nastopajo predvsem objekti abstraktne narave, zaradi kompleksnosti problemov pa smo največkrat prisiljeni razmišljati bolj ali manj abstraktno.

Koncepti abstrakcije so:

⇒ *Klasifikacija*
⇒ *Generalizacija*
⇒ *Agregacija*
  – *kartezična agregacija*
  – *agregacija na ravni objektov*
⇒ *Asociacija*

Klasifikacija je koncept abstrakcije, pri katerem se primerkom entitet, ki imajo neko skupno lastnost, priredi tip entitete, ki odraža to lastnost.
Janez, Bojan in Ivan so primerki entitete, tipa entitete študent. Njihova skupna lastnost je, da so študentje in tip entitete odraža to lastnost. Vsi trije bi lahko imeli tudi drugo skupno lastnost, npr. občani, športniki, itd.
Klasifikacija se nanaša na entitete in njihovo razvrščanje v tipe entitet

Generalizacija je koncept abstrakcije, ki je na višji ravni kot klasifikacija. Namreč pri generalizaciji množici tipov entitet, ki imajo neko skupno lastnost, priredimo splošni tip entitete, ki odraža skupno lastnost.

Vsem trem tipom entitete je skupna lastnost, da so občani in to je splošni, generalnizirani tip entitete, ki odraža to skupno lastnost elementarnih tipov entitete. Generalizacija nastopa na

višji ravni in opredeljuje razmerje med elementarnimi tipi entitet in posplošenimi tipi entitet, ki odražajo skupno lastnost.
Ko gradimo podatkovni model, med seboj kombiniramo koncepte.

Agregacija se pojavlja v celi vrsti modelov. Agregacijo ločimo na dve vrsti in sicer:

- kartezična agregacija
- agregacija na ravni objektov

Pri **kartezični agregaciji** gre za to, da množici atributov priredimo ustrezen tip entitete. Atributi: ime in priimek, naslov, spol, rojstni datum, … so del tipa entitete »študent«.

Agregacijo na ravni objektov uporabimo, kadar hočemo predstaviti strukturo nekega sestavljenega objekta (prostor, avto, državni organ, državna uprava, ….).
Agregacija in generalizacija se lahko uporabljata komplementarno.

To so trije karakteristični tipi abstrakcije, ki se nam pojavljajo v podatkovnih modelih. Tretji – agregacija – se pojavlja tudi v postopkovnih modelih.

Asociacija je oblika abstrakcije podatkov, pri kateri se podmnožici primerkov nekega tipa priredi na višjem nivoju asociirani tip, ki predstavlja skupne lastnosti asociiranih primerkov.

Razvoj podatkovnih modelov

Razvoj podatkovnih modelov poteka že desetletja in sicer od razvoja prvih računalnikov. Takrat so se najprej začeli razvijati **enostavni modeli**. Ti modeli so se ohranili vse do danes. Omogočajo nam organizacijo podatkov po datotekah. Takšna organizacija podatkov je bila na voljo do 70-ih let.

V začetku 70-ih let pa so se začeli pojavljati **izvedbeni modeli**. Podatki so lahko organizirani v datoteke (pisni dokumenti, manjše količine, 1 uporabnik) in podatkovne baze (večje organizacije, kompliciranе strukture, rabimo zahtevna orodja).

Izvedbene modele delimo na:

- *hierarhični model*
- *mrežni model*
- *relacijski model*
- *objektno orientirani model (90-ta leta)*

Izvedbene modele uporabljamo pri gradnji IS oz. pri gradnji baze podatkov, na katerem določen IS temelji. Kasneje se je skozi izkušnje pokazalo, da so ti izvedbeni modeli sicer nujni in uporabni v fazi izvedbe IS in izgradnje njegove baze podatkov, da pa niso najbolj primerni v zgodnejših fazah, ko se IS šele načrtuje. Zato se je vzporedno začel razvijati nov tip podatkovnih modelov - semantični model.

Semantični modeli se uporabljajo predvsem v fazi načrtovanja IS oz. njegove baze podatkov. Niso pa uporabni v fazi njegove izvedbe. Poudarek je na pomenu in vlogi podatkov. Eden prvih takih modelov je:

- *E-R model (entity-relationship)*
- *UML (universal modelling language)*

MODEL »ENTITETA – POVEZAVA«

E-R model je eden osrednjih semantičnih modelov, ki se uporablja pri načrtovanju IS. Angleško Entity relationship ali **E-R model**.

Semantični modeli so se začeli razvijati na osnovi ugotovitev iz praktičnih izkušenj, ki so pokazale, da t.i. izvedbeni modeli, ki se uporabljajo v nadaljnjih fazah izgradnje in izvedbe nekega IS oz. njegove baze podatkov, niso najbolj primerni v zgodnejši fazi razvoja IS, v fazi načrtovanja.

V fazi načrtovanja se osredotočamo na logične, vsebinske, funkcionalne vidike, ne pa na izvedbene in v teh zgodnjih fazah nas zanima predvsem pomen in vloga, ki jo imajo podatki v okviru načrtovanega sistema.

Eden prvih semantičnih modelov, ki se je v praksi uveljavil (za modeliranje podatkov na logični ravni) in je uporaben le v fazi načrtovanja IS, je **E-R model**. Nastal je v drugi polovici 70-tih let in se v praksi zelo hitro uveljavil ter ohranil vse do danes. Res pa, da ni bil nikoli standardiziran, zato ga različni avtorji predstavljajo na različne načine. Tako se grafična predstavitev od avtorja do avtorja razlikuje.

E-R model ali model entiteta-povezava temelji na uporabi treh temeljnih konceptov:

- *entiteta*
- *atribut*
- *povezava*

E-R model omogoča uporabo treh konceptov abstrakcije, in sicer klasifikacije, generalizacije in agregacije.

Entitete → entiteta je nekaj, kar je ali obstaja v realnem svetu ali v naših predstavah in je pomembno za obravnavani IS. Entiteta je lahko neki objekt oz. subjekt, ki fizično obstaja (vozilo, hiša, državljan,…), lahko pa je organizacijski ali nek drug konceptualni pojem (podjetje, oddelek, seminar,…).

Vsaka entiteta ima določene lastnosti, ki jih imenujemo atributi. Atributi opisujejo oz. določajo lastnosti entitet (entiteto državljan opisujejo atributi: priimek, ime, naslov, spol,…).

Tip entitete predstavljajo torej primerki z enakimi lastnostmi (tip študent predstavlja primerke; Ana, Marko, Ivo,…). Običajno imamo v okviru poljubnega IS opravka z množico tipov entitet.

Atributi → Vsaka entiteta ima neko lastnost in te lastnost opredeljujemo z atributi. Število atributov, ki opisujejo določeno entiteto, je poljubno.

Elementarni atributi so tisti, katerih vrednosti so nerazstavljive, medtem ko so **sestavljeni** atributi tisti, katerih vrednosti so razstavljive in jih lahko razstavimo na elementarne vrednosti.

Vsak atribut zavzame svojo vrednost, to pa je lahko:

- ena sama vrednost - **enovrednostni atribut** (državljan ima eno samo vrednost za atribut, starost ali rojstni datum)
- lahko pa je vrednosti več - **večvrednostni atribut** (atribut poklic pa lahko pri posameznih primerkih zavzame več vrednosti – oseba ima lahko več poklicev).

Domena atributa je množica vseh vrednosti, ki jih nek atribut lahko zavzame. Npr. domena atributa »ime« je množica vseh imen, domena atributa »poklic« je množica vseh poklicev, domena atributa »spol« je množica vseh spolov, ki pa ima samo dva elementa.

Zelo pomembna lastnost atributov so **ključni atributi**. V množici atributov, ki pripadajo nekemu tipu E, igrajo nekateri atributi nekoliko drugačno oz. posebno vlogo in to so ključni atributi oz. ključi. Ključni atributi se uporabljajo za:
- *enolično identifikacijo primerkov entitet,*
- *iskanje podatkov o primerkih entitet,*
- *vzpostavljanje povezav med primerki različnih tipov entitet.*

Glede na to razlikujemo med štirimi vrstami ključev.
- *primarni ključ*
- *sekundarni ključ*
- *tuji ključi*
- *speti ključi*

Vsaka entiteta ima praviloma atribut, katerega vrednosti omogočajo enolično identifikacijo posameznih primerkov entitet. Tak atribut imenujemo **ključni atribut oz. primarni ključ**. Kdaj je načelu enolične identifikacije zadoščeno?
- Pravilo 1: Pri podeljevanju primarnega ključa je treba paziti, da en primarni ključ pripada vedno enemu samemu primerku tipa entitete.
- Pravilo 2: En primerek tipa entitete sme imeti en sam primarni ključ. Vedno je potrebno preveriti, če je primarni ključ že v bazi podatkov.

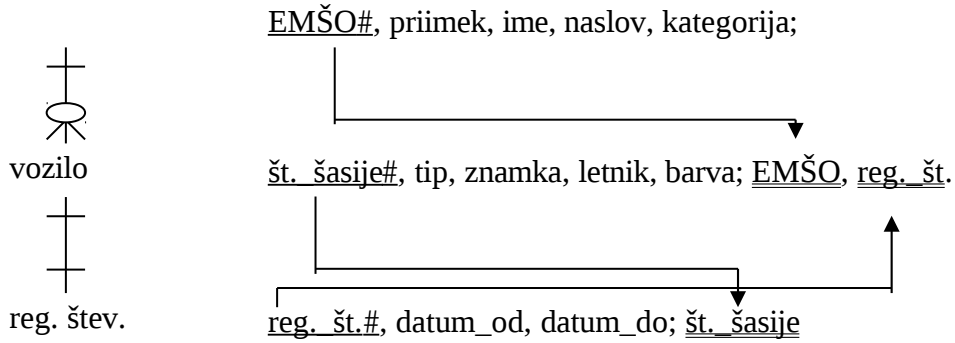
Primarni ključ je vedno en sam, tudi če imamo več atributov, ki bi izpolnjevali pogoje za primarni ključ.
Primarni ključ torej omogoča enolično identifikacijo primerka entitete in je zato tudi idealen instrument za iskanje podatkov o primerku entitete. Najpogosteje iščemo podatke po primarnem ključu. Kadar iščemo podatke po primarnem ključu, bomo vedno dobili natančno en zadetek. Iskanje po primernem ključu je najbolj pogosto in najučinkovitejše. Vendar pa se v praksi izkaže, da primerni ključ za iskanje ni dovolj.

Pogosto ga ne vemo, zato je koristno imeti še druge atribute, ki jih uporabljamo pri iskanju, in tem pravimo **sekundarni ključ**. Sekundarni ključi se uporabljajo za iskanje podatkov primerkov entitet. Pri tem pogoj enoličnosti ne velja več. Kot sekundarni ključ nam lahko nastopa pravzaprav vsak atribut določenega tipa entitete, z izjemo primarnega ključa. Ko gre za osebe, je v praksi najpogostejši sekundarni ključ »priimek in ime«. Ko iščemo podatke po sekundarnem ključu, običajno ne bomo dobili enega samega zadetka, ampak več zadetkov, kar lahko oteži iskanje.

Tretji tip atributa, ki igra posebno vlogo, je **tuji ključ.** Gre za atribut pri posameznem tipu entitete, s pomočjo katerega vzpostavljamo povezave z drugimi primerki tipov entitet.

Ilustracija problema, ki ga rešujemo s pomočjo tujih ključev, na konkretnem primeru, ko imamo tip entitete »vozilo«, ki ima atributa: lastnik in registrska številka.

lastnik



Tuji ključi nam služijo za vzpostavljanje povezav med primerki različnih tipov entitet.

Pri določanju tujih ključev veljajo določena pravila, ki so vezana na kardinalnost.
→ Vgradnja povezav s pomočjo tujih ključev, ko je **kardinalnost 1:1**
Pri kardinalnosti 1:1 vzpostavimo povezavo med tipoma entitet tako, da lahko vgradimo tuji ključ na eni ali na drugi strani. V primeru 1: 1 je vseeno na katero stran vgradimo tuji ključ, ki omogoča povezavo.
→ Vgradnja povezav s pomočjo tujih ključev, ko je **kardinalnost 1:N**
V tem primeru je tuji ključ vedno na tisti strani, kjer je kardinalnost N (več).
→ Vgradnja povezav s pomočjo tujih ključev, ko je **kardinalnost N:M (več:več)**
Takega primera ne moremo rešiti direktno (z vgradnjo tujega ključa na drugo stran). Vse povezave N:M ali M:N moramo razstaviti na dve povezavi 1:N. To opravimo s pomočjo t.i. presečne entitete, ki nam opredeljuje razmerje med tema dvema tipoma entitet. Na ta način se razmerje pretvori v 2 razmerji 1:N. Vgraditi moramo presečno entiteto, od tu naprej pa je postopek enak kot pri kardinalnosti 1:N. Pri presečni entiteti dobimo speti (sestavljeni) tuji ključ, ki je sestavljen iz dveh primarnih ključev. Hkrati je speti tuji ključ primarni ključ presečne entitete.

Speti ključi so vezani predvsem na primarne ključe. Idealno je, če ima določen tip entitete nek naraven atribut, ki omogoča enolično identifikacijo primerkov. Vendar se v praksi izkaže, da je teh primerov relativno malo. Največkrat moramo za to, da zagotovimo enolično identifikacijo primerkov, vpeljati umetni atribut. Včasih se lahko izognemo vpeljavi umetnega atributa pri določanju primarnega ključa tako, da pridemo do primarnega ključa s sestavljanjem več naravnih atributov in takemu ključu rečemo speti ključ, ki zagotavlja enolično identifikacijo. Ti atributi niso vedno naravni, včasih je lahko med njimi tudi kakšen umeten. Speti ključ je običajno primarni ključ, ki se sestoji iz več atributov.

Povezave – opredeljujejo odnose med tipi entitet v podatkovnem modelu. Povezava je koncept s katerim opredelimo razmerja med tipi entitet in njihovimi primerki. Tako kot pri entitetah, tudi pri povezavah ločimo med:

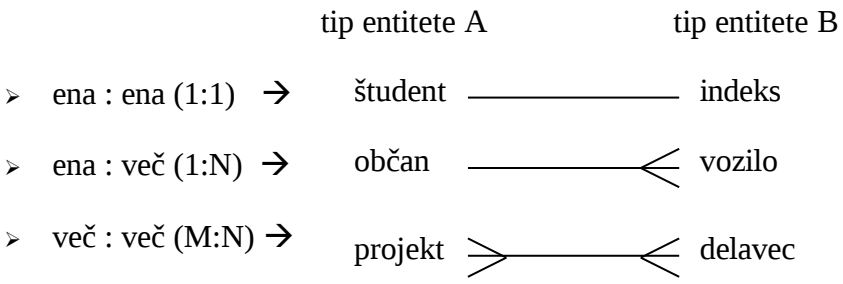
- **tipi povezav** in
- **primerki povezav.**

Tip povezave predstavlja množico primerkov povezav med primerki entitet. Vsak primerek povezave vključuje natančno en primerek tipa entitete, ki participira v povezavi.

Povezava »dela za« med tipoma entitet »zaposleni« in »oddelek«
Med tema tipoma entitet velja povezava »dela za«. Ta povezava nam opredeljuje razmerje med primerki tipa »zaposleni« in »oddelek«. Pri snovanju sistema moramo ta razmerja opredeliti in vsaka povezava mora imeti svoje ime. Tako imamo **ime povezave**, ki opredeljuje naravo razmerja med dvema ali več tipi oz. primerki entitet.

Naslednja lastnost povezav pa je **stopnja povezave**. Stopnja tipa povezave je določena s številom tipov entitet, ki so udeleženi v povezavi. Te so lahko binarne, kar pomeni, da gre za povezavo med primerki dveh tipov entitet in te so najbolj pogoste. Poznamo pa tudi povezave višjih stopenj, 3. in 4. stopnje, kjer gre za povezave treh ali štirih tipov entitet. Teoretično so lahko povezave poljubne stopnje, vendar pa v praksi najpogosteje nastopajo binarne povezave.

Kardinalnost oz. številnost povezave opredeljuje število primerkov povezav, v katerih lahko neki primerek entitete sodeluje. V bistvu ločimo tri tipe oz. vrste kardinalnosti:



Imamo tip entitete A, ki je v razmerju s tipom entitete B.

- Kardinalnost ena : ena (1:1) pomeni: en primerek tipa entitete A je povezan z natančno enim primerkom tipa entitete B in obratno, en primerek tipa entitete B je povezan z natančno enim primerkom tipa entitete A – simetričen.
- Kardinalnost ena : več (1:N) pomeni: en primerek tipa entitete A je povezan z enim ali več primerki tipa entitete B in en primerek tipa entitete B je povezan z enim samim primerkom tipa entitete A – asimetričen.
- Kardinalnost več : več (M:N) pomeni: en primerek tipa entitete A je povezan z enim ali več primerki tipa entitete B in obratno, en primerek tipa entitete B je povezan z enim ali več primerki tipa entitete A – simetričen.

Obveznost / neobveznost – povezave so lahko obvezne ali neobvezne. Če je povezava obvezna, mora vedno in v vsakem trenutku obstajati vsaj en primerek te povezave oz. primerek entitete, ki sodeluje v tej povezavi. Če pa je povezava neobvezna, pa pomeni, da je primerek entitete A lahko povezan s primerkom entitete B, vendar ni nujno.

Razvoj E-R modela

Koraki pri razvoju E-R modela → Ko razvijamo IS za neko konkretno poslovno področje, moramo to poslovno področje zelo podrobno in natančno analizirati. V okviru analize poslovnega sistema evidentiramo poslovne postopke in ugotavljamo, katere informacije so potrebne za izvajanje teh postopkov oz. ugotavljamo informacijske potrebe obravnavanega poslovnega področja. Te informacijske potrebe so vezane na določene ključne tipe entitet, na katere se informacije vežejo. Prvi korak v tem razvoju je identifikacija tipov entitet, določimo njihove atribute, opredelimo razmerja med temi tipi entitet in nato lahko začnemo z risanjem modela. Na koncu pa opredelimo še lastnosti povezav teh razmerij. Osnova je v analizi obravnavanega sistema in v analizi poslovnih procesov in postopkov, ki se izvajajo. Vzporedno s tem nastaja tudi postopkovni model.

Dokumentiranje E-R modela

- ***Grafični del*** (E-R graf) – iz tega grafičnega modela lahko razberemo tipe entitet, ki nastopajo v obravnavanem sistemu, povezave med njimi in pa lastnosti teh povezav. Nič pa

ne moremo razbrati o atributih in njihovih lastnostih. Zato običajno samo grafična predstavitev E-R modela ni dovolj.

- ***Opisni del*** – grafični del je potrebno dodatno dokumentirati z opisnim delom, ki mu pravimo **podatkovni slovar**. V tem opisnem delu so vse lastnosti podatkov, ki so v okviru obravnavnega IS podrobno predstavljene. Podatkovni slovar se običajno sestoji iz treh delov:
  - *slovar entitet*
  - *slovar atributov*
  - *slovar povezav*

Slovar entitet je dvodimenzionalna tabela, ki nam za vsak tip entitete opredeljuje seznam njegovih atributov ter vloge teh atributov.
Običajno ima slovar entitet tri karakteristične kolone:

- *oznako tipa entitete,*
- *naziv entitete in*
- *seznam vseh atributov, ki opisujejo lastnosti določenega tipa entitete.*

Pri pisanju tega seznama se držimo določenih pravil. Na prvem mestu običajno navedemo tisti atribut, ki igra vlogo primarnega ključa. Zaporedje ostalih atributov naprej ni posebej določeno. Vsi atributi, ki igrajo vlogo ključev so posebej predstavljeni. Pod tabelo je potrebno dodati legendo in v njej pojasniti, kaj kakšna stvar pomeni. Atribut, ki igra vlogo primarnega ključa # je podčrtan in z # oznako. Sekundarni ključi so samo podčrtani. Speti ključi so med seboj povezani s + (to pomeni, da je en ključ sestavljen iz več delov oz. atributov), tuji ključi pa so dvakrat podčrtani.

Slovar atributov nam za vsak tip entitete navaja seznam njegovih atributov in lastnosti oz. karakteristike teh atributov.
To je dvodimenzionalna tabela, predstavljena za vsak tip entitete posebej in vsebuje seznam njegovih atributov in njihove lastnosti. Vsak atribut mora imeti:

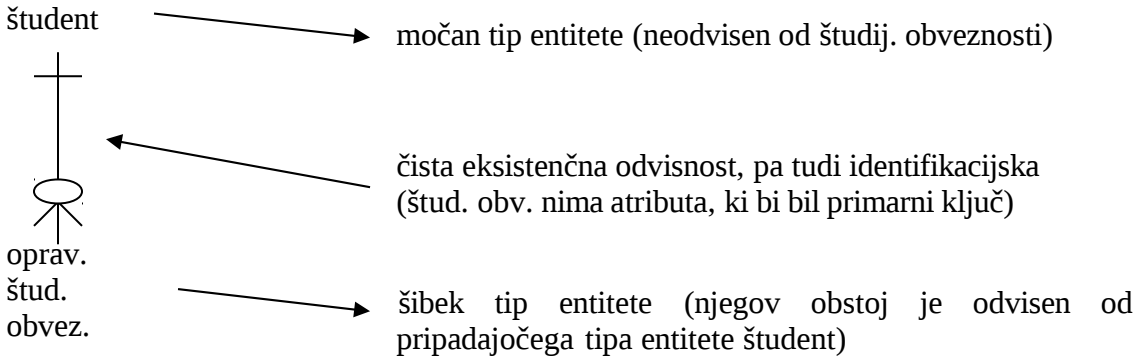
- *enolično oznako atributa* (A-01, ...);
- *opisno ime atributa*, ki mora čimbolj jasno povedati, za kateri atribut gre oz. kakšna je njegova vsebina;
- *standardno oz. skrajšano ime atributa* (običajno ne daljše od nepretrganega niza 8 znakov, izogibamo se šumnikom, če je besed več, jih povežemo z vezajem);
- *tip atributa*, ki nam pove, kakšne vrednosti lahko določen atribut zavzame – pri tem ločimo 3 temeljne vrednosti:
  - N so numerični atributi (med vrednostmi posameznih entitet lahko nastopijo samo številke);
  - A so alfabetski atributi (med vrednostmi posameznih entitet lahko nastopijo samo črke);
  - AN dopušča alfanumerične vrednosti – smisel določanja tipa atributa je v tem, da na osnovi tega opisa programerji programirajo vnosne maske za te podatke. Če je nek vnos samo numeričen, sprejme samo številke in na ta način se zmanjša možnost napak pri vnosu podatkov – vgrajena kontrola;
- *dolžina atributa* opredeljuje maksimalno dolžino nizov znakov, ki jo nek atribut lahko zavzame; z dolžino definiramo prostor na medijih, kamor shranimo podatke
- *standardne vrednosti* – ta kolona je lahko zelo koristna, vpišemo jo pri atributih, kjer je to le mogoče (vgrajevanje kontrol v računalniškem programu – lahko določimo 2 vrednosti M in Ž)

Slovar povezav opisuje lastnosti vseh povezav, ki nastopajo v okviru obravnavanega modela.
Je dvodimenzionalna tabela ki ima običajno naslednje kolone:

- *Oznaka povezave* – pomembno je, da je enolična, sicer pa ni predpisana

- *Povezane entitete* - v tej koloni opredelimo imena tipov entitet, ki so med seboj povezani. Npr. študent:štud._program, ...
- *Ime povezave* in običajno navedemo še
- *Kardinalnost* – pri kardinalnosti se poslužujemo naslednjega načina zapisa: NO:1O, 1O:NN in 1O:NO Opredeliti moramo kardinalnost oz. številnost in obveznost oz. neobveznost povezave. Prvi znak pomeni številnost oz. kardinalnost, drugi pa obveznost ali neobvezna.

Močni in šibki tipi entitet
Tipe entitet ločimo na t.i. močne in šibke tipe. Za močne tipe entitet velja, da je obstoj njihovih primerkov neodvisen od obstoja primerkov nekega drugega tipa entitete, s katerim je ta tip povezan.



Tip entitete študent lahko obstaja, tudi če ne obstaja noben primerek tipa entitete opravljena študijska obveznost. Zato je to močan tip.
Tip entitete opravljena študijska obveznost pa ne more obstajati brez tipa študent (ne more biti izpita brez študenta). Zato je to šibek tip.
Za šibke tipe entitet velja, da je njihov obstoj odvisen od obstoja primerkov entitet, s katerimi je ta povezan. Ta **odvisnost** pa je lahko:
- *eksistenčna*
- *identifikacijska*

Eksistenčna odvisnost → opravljena študijska obveznost ne more obstajati brez študenta.
Identifikacijska odvisnost nastopi takrat, kadar primerki nekega šibkega tipa entitete, nimajo lastnega atributa, ki bi zagotavljal njihovo enolično identifikacijo. Zato pogosto povzamemo kar primarni ključ močnega tipa entitete, s katerim je šibki tip povezan.

Slabosti E-R modela

Moč E-R modela je predvsem v tem, da ni usmerjen niti v izvedbo baze podatkov niti v predstavitev baze podatkov, ampak v konceptualno predstavitev podatkov, kakor jih vidi uporabnik.

Zaradi neenotnega standarda glede pomena njegovih osnovnih konceptov in glede grafične predstavitve E-R diagramov, pa ima predvsem naslednje slabosti:
- *ne omogoča povsem enotnega pogleda na podatke,*
- *ne omogoča povsem enotnega modelirnega postopka,*
- *ni univerzalni model, ki bi podpiral vse faze modeliranja podatkov,*

- *od konceptov abstrakcije omogoča le klasifikacijo, agregacijo in z razširitvami generalizacijo,*
- *ni standardiziran niti v pogledu uporabljenih konceptov niti glede grafične notacije.*

IZVEDBENI PODATKOVNI MODELI

Semantični modeli (E-R) se uporabljajo v fazi načrtovanja, ne pa v fazi izvedbe. Za samo izvedbo ta semantični model ni uporaben. Zato za izvedbo uporabljamo izvedbene modele.

Čeprav se izvedbeni modeli razvijajo že dobrih 30 let, danes še vedno govorimo o treh glavnih izvedbenih podatkovnih modelih:
- *hierarhični model*
- *mrežni model*
- *relacijski model (danes prevladujoč)*
- *objektno orientirani modeli*

Vsi ti modeli so nastali skoraj istočasno okrog leta 70. Njihova nadaljnja usoda pa je bila zelo različna.

Podatki v okviru IS so med seboj povezani in te povezave so zelo pomembne za delovanje IS. Če smo kakšno od teh povezav spregledali, bomo imeli kasneje pri uporabi tega IS probleme. Kajti to, kar je v model vgrajeno, bo dejansko tudi obstajalo. Vse kar smo pozabili, teh podatkov v IS preprosto ne bo, ali pa do njih ne bomo mogli priti, če ne bo ustreznih povezav.

Podatkovni modeli določajo način, kako so podatki med seboj povezani, strukturo podatkov in s tem tudi s pomočjo katerih ključev, atributov do podatkov pridemo. Zato moramo vsak IS zgraditi čim boljše in fleksibilno, kajti v praksi se pokaže, da zelo pogosto ključ, ki smo ga predvideli kot najpomembnejšega, ni znan. Potrebno je imeti vgrajene mehanizme, ki nam omogočajo vstop v bazo podatkov tudi po drugih kriterijih, ne samo po primarnem ključu. O tem nam pravzaprav odločajo podatkovni modeli: na kakšen način bomo do podatkov prišli in na kakšen način se bodo podatki obdelovali.

Hierarhični model je eden prvih resnejših, širše uporabljenih izvedbenih podatkovnih modelov. Nastal je okrog leta 1970 in sicer na osnovi praktičnih izkušenj in ugotovitev, da je realni svet, ki ga v okviru IS modeliramo, praviloma hierarhično strukturiran. Tem hierarhijam sledimo na vseh področjih realnega sveta, posebej pa še v organizacijah in poslovnih sistemih. Zato je hierarhični model zelo prikladen za modeliranje in obdelavo podatkov, ki imajo že po naravi hierarhično ali drevesno strukturo.

Glavna vstopna točka t.i. koren je na vrhu tega drevesa. Model se sestoji iz vozlič, ki prestavljajo posamezen zapis podatkov in povezav med vozlišči. Povezave so sestavni del te strukture in so vnaprej določene.

V hierarhičnem modelu nastopata dva temeljna modelirna koncepta – zapis ter povezava oče-sin. Zapis je zbirka polj, ki vsebuje podatke o primerkih entitet ali povezav med njimi. Povezava oče-sin je povezava tipa 1:N med dvema tipoma zapisov. En primerek povezave oče-sin se torej sestoji iz enega primerka zapisa tipa oče ter več primerkov zapisa tipa sin.

Grafično se hierarhična shema prikazuje kot hierarhični diagram, v katerem so tipi zapisov predstavljeni kot pravokotniki, povezave oče-sin pa kot daljice, ki povezujejo zapis tipa oče z zapisom tipa sin.

Hierarhična shema ima naslednje lastnosti:

1. na najvišjem nivoju imamo lahko en sam zapis, ki se imenuje koren, ta pa nima nadrejenih zapisov tipa oče
2. vsak tip zapisa, z izjemo korena, participira kot sin v natanko eni povezavi (sinovi imajo enega samega očeta)
3. vsak tip zapisa lahko participira kot oče v poljubnem številu povezav (oče ima lahko več sinov)
4. tipi zapisov v hierarhiji, ki ne participirajo v nobeni povezavi kot očetje, se imenujejo listi

Pri hierarhičnem modelu so poti iskanja oz. dostopa do podatkov vnaprej natančno določene. To nam omogoča zelo učinkovit dostop do podatkov, kadar iščemo podatke na tak način, kot je bilo v začetku predvideno. Problem pa nastopi takrat, ko bi radi prišli do podatkov na nek drug način. V tem primeru tak model postane izrazito počasen. To je njegova glavna slabost – povezave so vgrajene, vnaprej določene, kasneje jih je težko spreminjati in to nam omejuje dostop do podatkov, kadar gre za dostop, ki je drugačen od tistega, ki je bil v začetku predviden.

To je najstarejši model, ki se je v praksi zelo hitro uveljavil in se dokaj široko uporabljal dolga leta in še danes ga srečamo na nekaterih specifičnih področjih, kjer je učinkovitost IS zelo pomembna, npr. v bankah, kjer se natančno vnaprej ve, na kakšen način se dostopa do podatkov; opravka imamo z veliko transakcijami, ki se vedno izvajajo na enak način).

Mrežni model je pravzaprav razširitev hierarhičnega modela. V praksi pa se je uveljavil v drugi polovici 70-tih let. Oba sta bila prevladujoča modela vse do začetka 90-let, ko je začel prevladovati relacijski model.

Mrežni model je razširitev hierarhičnega v tem smislu, da dve temeljni pravili, ki veljata za hierarhični model, ne veljata več. To pomeni, da imamo lahko na najvišjem nivoju tudi več zapisov (ne samo enega) in vsak zapis ima v tej hierarhiji poljubno število podrejenih in poljubno število nadrejenih zapisov. To pomeni, da imamo lahko več vstopnih poti v to strukturo, kar bi praktično pomenilo, da lahko pridemo do podatkov tudi po drugi poti: npr. ne samo po vpisni številki, ampak tudi preko priimka, rojstnega datuma ali kakšnega drugega atributa. - To so prednosti tega modela. Ena temeljnih slabosti je enaka kot pri hierarhičnem modelu – povezave med podatki so vnaprej določene, vgrajene v model, kar nas kasneje utesnjuje in omejuje. Ko razvijamo nek IS, vnaprej ne poznamo vseh informacijskih potreb, zato imamo lahko kasneje težave pri teh dveh modelih.

Te slabosti odpravlja **relacijski model**, ki je zasnovan precej drugače od prvih dveh modelov. Čeprav je nastal praktično istočasno, se je začel uporabljati šele sredi 90-tih let. Danes je v praksi prevladujoč in najbolj široko uporabljan podatkovni model.

Gre za model, ki ima edini od teh treh, matematično podlago. Pravila njegove zasnove so matematično določena, utemeljena na t.i. relacijski teoriji. Relacijski model se sestoji iz relacij. Relacija pa je v tem modelu relacijska dvodimenzionalna tabela, ki vsebuje podatke praviloma enega tipa entitete. Tabele so izoblikovane okrog tipa entitet. Praviloma za vsak tip entitete, ki nastopa v E-R modelu, dobimo eno ali več relacijskih tabel.

Relacijska tabela je dvodimenzionalna, katerih kolone določajo število atributov, vrstice pa število primerkov**.** V okviru vsakega IS nastopa več tipov entitet, tako dobimo tudi več takih tabel. Torej ta model spremlja množica relacijskih tabel (tudi do nekaj tisoč).

V primerjavi s prejšnjima dvema modeloma med temi tabelami ni nobenih povezav. Povezav med tabelami ne predvidevamo vnaprej. To je najpomembnejša razlika. Pri relacijskem modelu se povezave med temi tabelami vzpostavljajo v realnem času (ko se podatki obdelujejo) na osnovi zahtev uporabnika. Ko uporabnik pove, katere informacije želi dobiti iz IS, računalnik ugotovi, katere relacije mora med seboj povezati, da bo lahko dal odgovor na zastavljeno vprašanje. Potem v realnem času vzpostavi povezave, izdela odgovor na vprašanje in ga posreduje uporabniku. Za vsako transakcijo se posebej vzpostavljajo povezave. To je glavna prednost tega modela, hkrati pa tudi slabost.

Model je zelo fleksibilen, saj lahko med podatki vzpostavimo poljubne povezave, ki nas ne utesnjuje z vnaprej določenimi povezavami kako bomo do podatkov prišli. Slabost tega modela se izkaže v realnem času, ko uporabnik postavi svoje zahteve, ima računalnik razmeroma veliko dela, da ugotovi, katere zadeve je potrebno med seboj povezati za to, da bo lahko odgovoril na vprašanje. To lahko zahteva na tisoče, tudi milijone operacij, kar lahko terja nekaj časa, pač odvisno od zmogljivosti računalnika. Zato je bil to v preteklosti problem in to je tudi razlog, zakaj se toliko časa ni uveljavil. Na nekaterih specifičnih področjih se ta model še vedno kombinira s hierarhičnim in mrežnim modelom, da dosežemo večjo učinkovitost, predvsem tam, kjer gre za velike baze podatkov, ko mora računalnik opraviti veliko število transakcij (banke, rezervacije letalskih vozovnic, ...)

PROBLEM PREHODA IZ LOGIČNE NA FIZIČNO RAVEN ZASNOVE IS

razvojne faze		pričakovani rezultati posamezne razvojne faze IS
1. faza – strateško načrtovanje >	model obravnavanega sistema	> opis funkcije OS > opis organiziranosti OS > opredelitev info. potreb > identifikacija postopkov
2. faza >	logični model IS	> opredeljene so vse vsebinske lastnosti sistema na logični ravni
3. faza >	fizični model IS	> opredeljene so vse lastnosti sistema na fizični ravni
4. faza >	izvedbeni model IS	> izdelana in preverjena info. rešitev

V 2. fazi so opredeljene vse vsebinske in funkcionalne lastnosti sistema. Razlika med logično in fizično ravnijo: logična raven naj bo še tehnološko, če ne povsem neodvisna, medtem ko fizična raven že pomeni upoštevanje vseh konkretnih izvedbenih elementov – izbranih orodij tehnologije.

Pregled uporabe metod in tehnik skozi razvojne faze IS za podatkovni in postopkovni vidik

slide 118

Pri modeliranju podatkov gre v 1. fazi za tekstualni opis, lahko pa izdelamo tudi že nek grobi E-R model, ki predstavlja glavne tipe entitet. V 2. fazi se naredi podroben E-R model (vse faze N:M pretvoriti v 1:N s presečnimi entitetami) in izdelamo podroben podatkovni slovar. Dejansko je podatkovni vidik sistema uobličen v podatkovnem slovarju. V 3. fazi preidemo na fizično raven. Pri tem je ena od temeljnih težav v tem, da E-R model ni uporaben na izvedbeni ravni, ni uporaben za izvedbo baze podatkov. Izvedbeni modeli so: relacijski, objektno orientirani, hierarhični in mrežni. Glavna naloga, ki nas tu čaka je, da E-R model pretvorimo v enega od štirih izvedbenih modelov. Danes je to običajno relacijski model. Ta pretvorba je lahko bolj ali manj kompliciranа. Zaključna, 4. faza je generirana baza podatkov na osnovi izbranega podatkovnega modela. S tem je na podatkovni ravni več ali manj postorjeno.
Na postopkovni ravni gre v 1. fazi za tekstualne opise, različne organigrame, ki nam kažejo organizacijsko strukturo. V 2. fazi sledi podrobno razdelan strukturni graf, da pridemo do elementarnih postopkov, ki morajo biti evidentirani in predstavljani; diagram toka podatkov, ki nam predstavljajo vse podatkovne tokove v okviru načrtovanega IS in pa sistema z okolico – dobimo zbirke podatkov in iz tega lahko naredimo povezavo z E-R modelom (ugotovimo če je pravilen). Tu obstaja še cela vrsta drugih tehnik, kot npr. prehodni diagrami (v knjigi), ….. V 3. fazi je potrebno na osnovi teh ugotovitev in opredeljenega postopkovnega modela na logični ravni podrobno opredeliti vse programske module, s katerimi se bodo postopki izvajali. Tu imamo na voljo spet celo vrsto tehnik. Potrebno je opredeliti vhodne in izhodne maske, opredeliti na kakšen način se bodo podatki zajemali. Nato se v 4. fazi se programira programska koda v izbranem programskem jeziku.

Ta prehod iz logične na fizično raven je lahko manj ali pa bolj zahtevna. To je deloma odvisno od tega, koliko smo se s temi vprašanji ukvarjali že v fazi izdelave logičnega modela. Preveč razmišljati vnaprej ni dobro, saj je izvedbeni model že odvisen od tehnologije in orodij, zato mora biti čimbolj neodvisen. **Sam postopek pretvorbe iz logičnega v fizični model imenujemo normalizacija**. Gre v bistvu za dokaj formaliziran postopek. V primeru, ko gre za pretvorbo E-R modela v relacijski model, je to formaliziran postopke. Pretvorbe so natančno določene. Kadar pa gre za pretvorbo E-R modela v nek drug model, ki ni tako strogo matematično formaliziran kot relacijski, potem ta pretvorba ni vnaprej tako natančno formalizirana. Postopek normalizacije ima 3 karakteristične korake (se ne bo spuščal v podrobnosti – v knjigi pa podrobno predstavljen).
Gre pa v bistvu za to, da naš E-R model preslikamo v relacijski model, kar pa praktično pomeni, da moramo preslikati naše tipe E v relacije. V relacijskem modelu so naši tipi E predstavljeni z dvodimenzionalnimi tabelami. To pa mora potekati v skladu z zelo natančno opredeljenimi pravili.

Pretvorba podatkovnega modela v tretjo normalno formo

slide 122

TEHNOLOŠKI, ORGANIZACIJSKI IN KADROVSKI VIDI RAZVOJA IS

Osredotočenje na opredelitev značilnosti okolja, ki je pomembno za razvoj informacijskih sistemov. Če želimo, da se neka dejavnost uspešno odvija in razvija, potem je zelo pomembno, v kakšnem okolju se to dogaja. Pod okoljem razumemo tehnološko okolje; orodja, ki jih imamo na voljo; tehnično infrastrukturo, ki jo imamo na voljo; pod okoljem razumemo organizacijske pogoje, kako je področje v celoti organizirano in pa kadrovski vidik, ki je tudi zelo pomemben. Vsi trije vidiki so med seboj prepleteni in predvsem drug od drugega soodvisni. Ni dovolj samo dobra tehnologija, če nimamo ljudi, ki bi jo uporabljali, ali če smo slabo organizirani. Gre za kompleks, ki je med seboj tesno povezan in soodvisen.

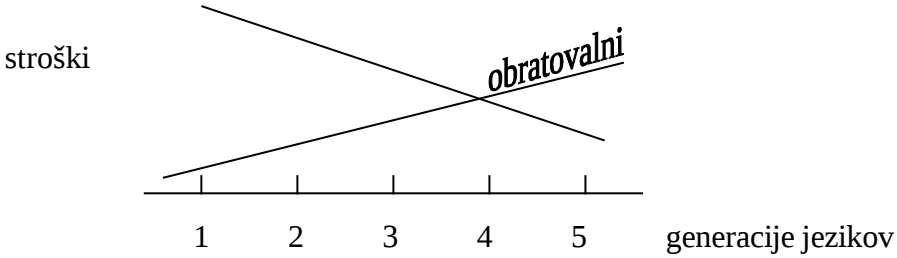
!!!! PIV!!!! Informacijska orodja

- jeziki četrte generacije
- krmilni sistemi podatkovnih baz
- slovarji podatkov
- CASE orodja

Informacijska orodja so ključnega pomena za učinkovit razvoj IS-ov. Teh orodij je danes na voljo zelo veliko (vsak program s katerim delamo, bi lahko šteli za eno od info. orodij). Predstavlja samo tista, ki so najpomembnejša in so ključnega pomena za uspešen razvoj IS.

Jeziki 4. generacije, počasi bi lahko dodali tudi že jezike 5. generacije. O programskih jezikih in njihovem razvoju že pri informatiki. Jeziki 4. generacije so prinesli pomemben zasuk na področju razvoja IS, predvsem v pogledu večje produktivnosti in skrajševanja razvojnih časov (ključni problem razvoja IS so tudi danes predolgi razvojni časi in visoki razvojni stroški). Glede na to, koliko časa bo trajal razvoj nekega IS in kakšna bo rešitev, na to v veliki meri vplivajo tudi druga orodja, zelo pomembno vlogo pri tem pa igrajo ravno programski jeziki, ki jih uporabljamo pri razvoju rešitev. Jeziki 4. generacije so v razvoju že 20 let, nadomestili naj bi 3. generacijo, vendar se to v vseh teh letih ni povsem zgodilo. Danes se jeziki 4. generacije uporabljajo vzporedno oz. komplementarno z jeziki 3. generacije. V to vstopa pa že tudi 5. generacije. Katero orodje bomo uporabili, je seveda odvisno od značilnosti in karakteristik IS, ki ga razvijamo – to ni vnaprej jasno določeno, niti ni to zgolj odvisno od tega, kako moderna orodja želimo. To mora biti zelo pretehtana odločitev, utemeljena na karakteristikah rešitve, ki jo razvijamo, pri čemer sta pomembna dva parametra, ki jim imamo vedno v prvem planu: čas, ki ga imamo na voljo za razvoj in stroški. Oboje je v veliki meri odvisno od izbora generacije programskih jezikov. Posebej pa je potrebno imeti pred očmi dve vrsti stroškov: razvojni in obratovalni oz. vzdrževalni stroški. Znotraj tega moramo optimizirati ene in druge stroške – vsoto teh stroškov pa moramo držati navzdol. Odvisno pa je od primera do primera: lahko so razvojni stroški zelo visoki, vzdrževalni pa so minimalni (npr. odmera davka – razvojni stroški visoki, obratovalni v smislu uporabe pa so majhni – odvrtijo se1 x letno), na drugi strani pa so obratovalni stroški lahko bistveno večji kot razvojni (npr. bančno poslovanje, register prebivalstva – 365 dni na leto). Torej je v tem primeru potrebno optimizirati predvsem obratovalne stroške. Generalno velja: čim višjo generacijo jezikov uporabimo, nižji bodo razvojni stroški. Jeziki 4G so v pogledu časa in razvojnih stroškov bistveno boljši kot 3G (razvojni čas je krajši, razvojni stroški nižji), toda rešitve, ki na ta način nastanejo, so običajno v obratovanju dražje.

razvojni



Generalno velja, da se razvojni stroški znižujejo z višjo generacijo jezikov, pri tem pa običajno potrebujemo več režije – zmogljivejše računalniške sisteme – zato se obratovalni stroški zvišujejo. Zato iščemo optimum.

!!! ZPIV!!! Izjemno pomembno orodje, brez katerega razvoj IS danes skorajda ni več mogoč, je **krmilni sistem baz podatkov**. Da bazo podatkov lahko zgradimo, potrebujemo ustrezna orodja, ki nam omogočajo strukturo baze podatkov (ta je odvisna od izbranega podatkovnega modela). Najprej se moramo odločiti za podatkovni model, npr. relacijski model. Ta nam omogoča strukturo podatkov. Nato pa potrebujemo orodja, ki nam bodo omogočala kreiranje baze podatkov, ažuriranje baze podatkov, uporabo podatkov, vzdrževanje baze podatkov in še celo vrsto drugih pomožnih funkcij, kot npr. zaščito podatkov. Krmilni sistem je sistem specializiranih programov, ki nam omogočajo vse te funkcije: strukturo, nadziranje, ažuriranje, uporabo, …. To so zahtevni in kompleksni programski sistemi. Kako priti do takega programskega sistema - ali ga razvijemo sami, ali ga kupimo? Danes ga običajno kupimo, saj je na trgu množica teh krmilnih sistemov baz podatkov. Lastnega razvoja se lotimo v kakšnih posebnih primerih, ko bi presodili, da takšnega produkta, ki bi našim zahtevam ustrezal, na trgu ni. Kar pa je zelo redko. Najbolj razširjen na trgu je Oracle, ki pokriva okrog 40% svetovnega trga; Infomix, Paradox, Access (ta bolj za PC).

Slovarji podatkov – že vemo, kaj vsebuje. Kako ga vzdrževati v neki večji organizaciji? Ta slovar podatkov je v večji organizaciji lahko zelo obsežen. Urejene organizacije imajo izdelan slovar podatkov za celoten poslovni sistem. To pomeni, da imamo v takem slovarju nekaj sto entitet in nekaj tisoč atributov. Ročno vzdrževanje tako obsežnega slovarja je nemogoče, zato potrebujemo zanj posebna orodja. Na trgu so se že pred mnogimi leti začela pojavljati posebna orodja za vzdrževanje slovarja. Izkušnje so pokazale, da samostojni produkti, ki niso povezani z drugimi razvojnimi orodji, niso najboljša rešitev. Zato je danes funkcija slovarja podatkov v glavnem že vključena v zbirko t.i. CASE orodij.

CASE orodja (Computer aided system engineering)

- opredelitev CASE orodij
- razvojne smeri
- vrste CASE orodij:
    - za vzdrževanje dokumentacije
    - za reinženiring
    - za podporo celotnemu razvojnemu ciklu
    - za podporo vodenju projekta
    - za izboljšanje kakovosti

Razvoj teh orodij se je prav tako začel pred 20 leti. Na začetku so bila ta orodja funkcijsko ozka. Omejevala so se v glavnem na grafiko (pomagala so nam risati nekaj diagramov: strukturni, E-R, DTP), nato pa se je funkcionalnost teh orodij razširila na vsa področja in v vse faze razvoja nekega IS, začenši s strateškim načrtovanjem, logično zasnovo, fizično zasnovo in seveda v samo izgradnjo, programiranje; in kasneje v spremljanje v fazi vzdrževanja

in v nadzor kakovosti. Vse to je vključeno v ta sklop. To so želje in cilj vsake večje organizacije – sistem, ki celovito pokriva vse funkcije IS. Žal kljub 20 letnim razvojnim prizadevanjem in velikemu napredku razvojnih orodij, pa orodij, ki bi celovito pokrivala vse razvojne aktivnosti, je na trgu še vedno zelo malo. Več je orodij, ki so specializirana na posamezne segmente tega razvojnega procesa in po posameznih sklopih in problemih, s katerimi se srečujemo v razvoju.

Eden od problemov, s katerimi se srečujemo na področju razvoja IS organizacije od samega začetka, je **vzdrževanje projektne dokumentacije**. Gre za razmeroma kompleksne projekte, katerih velik del je abstraktne narave. Če nimamo dokumentacije, potem je praktično v množici programov, na katerih temelji delovanje IS, nemogoče iskati tisti del, ki bi ga radi spremenili za to, da bi nekaj želeli spremeniti. Če sistem ni dobro dokumentiran, je to praktično nemogoče. Zato je projektna dokumentacija izredno pomembna. Pomembnosti dokumentacije se zavemo šele takrat, ko nastopi problem. Za vzdrževanje dokumentacije so ta orodja odlična, ker so narejena tako, da je dokumentacija projekta stranski produkt razvojnega procesa. CASE orodja morajo imeti funkcijo, ki nam omogoča vzdrževanje dokumentacije, da se bo vsaka sprememba, ki jo naredimo v sistemu, avtomatično dokumentirala tudi v projektni dokumentaciji in v slovarju podatkov, če se nanašajo na podatke. S tem pa je povezan tudi izbor ustreznih orodij.

Orodja za reinženiring – ta orodja so bila nekaj let nazaj (1-3) izjemno aktualna. Ideja pri teh orodjih je ta, da bi z njihovo pomočjo rešitve, ki so nastale že pred mnogimi leti, pa so vsebinsko in logično še povsem dobre, vendar pa tehnološko zastarele, na relativno enostaven način posodobili. V velikih organizacijah (tudi pri nas nekaj takih: ZPIZ, zdravstveni zavod, nekatera ministrstva, ki so svoje info. rešitve začele razvijati v 70-tih letih) mnoge rešitve že dolga leta tečejo v skoraj nespremenjeni obliki. V določenem trenutku pa se izkaže, da so take rešitve vsebinsko še dobre, so pa tehnološko zastarele – zapisane v nekem »obskurnem« jeziku 3. generacije. Kako to rešitev, v katero je bilo vloženo ogromno denarja, ohraniti s čim manjšimi stroški? Zato so v 90-tih letih začeli razvijati posebna orodja, ki so narejena tako, da znajo iz stare programske kode narediti logično sliko sistema in to logično sliko preslikati v fizični model, iz tega pa generirati rešitev za današnjo tehnologijo. To so zelo specializirana orodja, ki so bila še posebej zelo v rabi npr. ob prehodu 2000/2001 in bodo ponovno, ko bo šel tehnološki razvoj naprej za eno generacijo.

Orodja za podporo celotnemu razvojnemu ciklu – to je pravzaprav osrednji namen večine teh orodij: razviti celoten razvojni cikel.
Orodja za podporo vodenju projektov, za nadzor kakovosti, za vzdrževanje kakovosti, in druga.
(CASE orodja so programska orodja, programski sistemi, informacijska orodja, ki nam olajšajo, čimbolj posplošijo, avtomatizirajo razvoj IS, sistematizirajo in omogočajo vzdrževanje dokumentacije, …..)

Organizacijski, tehnološki in kadrovski vidiki razvoja informacijske infrastrukture

- dolgoročno načrtovanje razvoja informacijske infrastrukture
- integracija tehnologij
- spremljanje tehnoloških razvojnih trendov
- naložbeni vidik
- organizacijski vidik

- kadrovski vidik

Če govorimo o tem, kakšno okolje je potrebno v naši organizaciji, da bo ta razvoj potekal čimbolj uspešno in skladno z našimi razvojnimi cilji, potem je temeljno izhodišče, da je potrebno ustvarjati tako okolje, ki je naravnano na dolgoročno načrtovanje razvoja inf. infrastrukture. Lahko se nam pojavi cela vrsta elementov in razvojnih trendov, ki jih moramo imeti pred očmi. Eden od razvojnih trendov, ki je pomemben z vidika dolgoročnosti, je integracija tehnologij, ki je po vseh večjih organizacijah zelo resen problem. V Upravi imamo v glavnem opraviti z zelo velikimi organizacijami, za katere je značilno, da so skozi desetletja razvoja tega področja razvile izolirane tehnološke otočke (po različnih izpostavah, …). Tako imamo neke lokalne rešitve, ki so med seboj slabo ali pa sploh nepovezljive. Naša dolgoročna razvojna vizija vseh sodobnih organizacij pa je razviti enovit enoličen informacijski sistem, enovito tehnološko podporo funkcijam celotne organizacije (ministrstva, celotne uprave). To pomeni, da je potrebno te izolirane tehnološke otočke med seboj čimbolj povezati, integrirati, kar pa ni lahko – brez enotnih standardov, imajo različno programsko, strojno opremo, jezike itd.

Drugo izhodišče dolgoročnega razvoja je spremljanje glavnih razvojnih tehnoloških trendov. To področje je razmeroma mlado in izredno pestro. Ponudnikov je ogromno in rešitve, ki nam jih ponujajo, med seboj niso vedno kompatibilne. Zato mora nekdo v organizaciji poznati razvojne tehnološke trende, postaviti usmeritve in vse razvojne projekte moramo temu prilagajati. To je izredno pomembno, ker so investicije za to področje postale izjemno veliko breme tudi za dobro stoječe organizacije.
Naložbeni vidik v preteklosti na tem področju ni igral vidne vloge. V preteklosti so predstavljale investicije v informacijsko tehnologijo majhen odstotek skupnih investicij. Danes pa so te investicije v IT vedno večje breme za organizacije, ker oprema naglo zastareva, zato je ta naložbeni vidik postal izjemno pomemben.

Tehnološki vidiki

- decentralizacija procesne moči
- cenejši in zmogljivejši računalniki
- omreženje računalnikov
- integriranje lokalnih in oddaljenih omrežij
- internet/intranet
- omrežno računalništvo

Eden temeljnih tehnoloških trendov, ki smo mu bili priča skoraj 10 let, je decentralizacija procesne moči, ki se je zgodila pri večini uporabnikov. Kaj to dejansko pomeni? V vseh večjih organizacijah se je v začetku 90-tih let prenos obdelav z velikih centralnih sistemov na manjše decentralizirane sisteme po organizacijah. Ta proces je že sredi 90-tih let povzročil, da smo imeli namesto enega velike centralnega sistema množico majhnih procesnih enot, ki so bile v začetku med seboj relativno slabo povezane, ali pa nepovezane. Glavni razlog za decentralizacijo je poleg pojava osebnih računalnikov tudi ugotovitev, da stane vzdrževanje informacijskih sistemov veliko več na enem velikem, kot pa na ustrezni množici manjših sistemov. Ugotovili so, da ena enota procesne moči (enota, da obdelamo določeno količino podatkov v določenem času) stane na centralnem sistemu približno 100 x toliko, kot na PC-ju. Če bi lahko poenostavili: če neko delo opravimo namesto na enem velikem sistemu na 20 PC, in če je razmerje v enoti 1:100, je to bistveno znižanje stroškov. Ta razmerja so usmerila organizacije, da so začele svoje računalniške obdelave in vzdrževanje sistemov prenašati s centralne na lokalno raven. Po nekaj letih, ko se je nabralo nekaj izkušenj in so se tudi ti stroški

sešteli, so ugotovili, da razmerje ni povsem realno. Prednost je sicer v tem, da posamezno obdelavo prenesemo na posamezna delovna mesta, kjer ima uporabnik nadzor nad obdelavo, sam vpliva na dela (kdaj bo podatke vnesel, kdaj bo naredil poročilo, ipd), hkrati pa se je pokazala tudi cela vrsta slabosti na drugih področjih. Če razmišljamo s stališča organizacije, se izkaže, da je vzdrževanje teh sistemov drago in problematično, pojavi se problem varovanja podatkov, itd. (sredi 90-tih let je decentralizacija napolnila malho Microsoftu in še nekaterim drugim podjetjem, ki je na ta način postal »monopolist« - vsaka nova verzija pomeni nove stroške, licence, preinštalacija, vzdrževanje tolikih računalnikov, …) Izkazalo se je, da je šel ta proces predaleč.

Kot odgovor na to se je pojavil koncept omrežnega računalnika, ki je v bistvu koncept »inteligentnega« terminala, kakršnega smo poznali v 70-tih letih v velikih organizacijah. To pomeni povratek k enemu močnejšemu, zmogljivejšemu sistemu »strežniku«, ki vzdržuje določeno mrežo delovnih postaj. Ti pa niso več tako razkošno opremljeni, niso več tako dragi, nimajo tolikšne zmogljivosti – imajo manjšo zmogljivost, so cenejši in manj problemov je z njihovim vzdrževanjem. Ta ideja se je pojavila okrog 96., 97. leta.

Drug trend, ki je zelo pomemben in poteka že 30 let, so cenejši in zmogljivejši računalniki. Trend nenehnega povečevanja procesne moči računalnikov se nadaljuje z nezmanjšanim tempom. T.i. Moorow zakon (ki je že koncem 70-tih let postavil enostavno formulo): procesna moč računalnikov se podvoji vsakih 18 mesecev. To še danes velja. Dejansko se moč računalnikov podvaja na 18 mesecev ob cenah, ki za isto enoto moči še vedno padajo.

Omreženje računalnikov je postalo dejstvo in je povezano z integracijo lokalnih in oddaljenih globalnih omrežij. Še nekaj let nazaj so bile povezave med tema dvema svetovoma zelo omejene. Danes pa imamo internet/intranet, ki nam omogočata uporabo istih orodij – tako znotraj poslovnega sistema (računovodstvo, knjigovodstvo), ali pa informacije, ki jih potrebujemo zunaj svojega okolja.
Omrežno računalništvo pa je eden možnih trendov za v prihodnje.

Organizacijski vidiki

- spremenjena vloga informacijske poslovne funkcije
- od avtomatizacije k informatizaciji
- od »AOP« centra k »štabni službi za informatiko«

Danes se je vloga informacijske poslovne funkcije zelo spremenila. Zgodil se je prehod od avtomatizacije, ki smo jo izvajali približno 20 let, k informatizaciji. Avtomatizacija je bila parcialna, osredotočena na preprosta rutinska opravila. Tehnološka podpora je bila takrat praviloma 1 centralni računalnik. Danes pa govorimo o informatizaciji, ki se zajeda v vse pore organizacije, na vsa delovna mesta, v vse hierarhične ravni, spreminja poslovne procese, spreminja organizacijske strukture, spreminja odgovornosti za izvajanje – podobno se je spremenila tudi vloga poslovne funkcije in je zadolžena za informatizacijo organizacije. Včasih je bila ena centralizirana poslovna funkcija, ki je bila zadolžena za vse aktivnosti, povezane z obdelavo informacij za organizacijo. Sedaj se je velik del teh funkcij, ki jih je včasih izvajal AOP center, direktno preneslo kar k uporabnikom: zajemanje podatkov, vsi teksti, …. Tudi samo procesiranje se je v veliki meri decentraliziralo. Zato danes potrebujemo drugačno službo, poslovno funkcijo. Danes je vmesno govoriti o »štabni službi za informatiko«, ki mora biti oz. običajno tudi je v organizacijski strukturi postavljena čim bližje vodstvu celotne organizacije.

Ta služba za informatiko ima danes povsem drugačno nalogo, kot nekdanji AOP center. Predvsem so to:

- strateško načrtovanje in razvoj IS, informacijske infrastrukture v organizaciji;
- izredno pomembna in obsežna naloga je postalo izobraževanje. Potrebno je sistematično skrbeti za znanja, ki jih potrebujemo na različnih ravneh. Nekdo mora skrbeti in ugotavljati potrebe, razvijati in iskati nujne ustrezne izobraževalne oblike usposabljanja in zagotavljati, da se to usposabljanje tudi izvaja;
- podpora uporabnikom, ki pri delu naletijo na določene probleme, mora biti ustrezno organizirana in zagotovljena;
- varovanje in zaščita podatkov je postal eden od osrednjih problemov, še posebno v Upravi, kjer imamo opravka z občutljivimi osebnimi podatki, ali z drugimi podatki, ki so vezani na varnost države
- standardi opreme in želje po integraciji tehnologij – poenotenju razvoja – nekdo mora postavljati minimalne standarde na področju strojne opreme, programske opreme, orodij, rešitev, itd.

To so glavne naloge štabne službe za informatiko, ne pa samo operativno izvajanje.

Kadrovski vidiki

Dejansko se da IS in info. rešitve razvijati, ter jih tudi uporabljati oz. vzdrževati, samo z ustrezno usposobljenimi ljudmi. Te ljudi imamo lahko znotraj organizacije, ali pa določena znanja kupujemo tudi na »trgu«. Dobro pa je vedeti, katera vsebinska področja je potrebno sploh pokrivati in katere sklope znanj je potrebno obvladovati. Za vsakega od teh sklopov se potem vprašati ali ga pokrivati z lastnimi kadri ali pogodbeno z zunanjimi. Takšni sklopi so:

Strateško načrtovanje in razvoj informacijske infrastrukture strukture – to je zelo zahtevno področje. Strokovnjakov za to je na trgu relativno malo. V večji organizaciji je dobro, da imamo za vitalne naloge svoje ljudi, ki to področje obvladujejo; za druge, ki so manj kritične ali časovno odvisne, pa lahko ljudi zunaj organizacije.

Razvoj IS – tu običajno potrebujemo znanja z večih področij. Na eni strani potrebujemo informacijska, na drugi strani pa funkcionalna znanja področja, za katerega IS razvijamo. Ta konkretna znanja lahko dobimo samo znotraj hiše, ki morajo o teh info. rešitvah nekaj vedeti in morajo aktivno sodelovati, da bi prišli do dobre rešitve. To je danes največji problem v večini organizacij (sodelovanje ljudi z obeh področij).

Uporaba informacijskih rešitev – ta znanja morajo biti znotraj hiše.

Uporaba informacijskih orodij – en del moramo vzdrževati znotraj hiše, drug del lahko damo projektirati ven.

Vzdrževanje strojne in programske opreme – deloma mora biti pokrita znotraj hiše. Zunanje vzdrževanje je navadno povezano z velikimi stroški, če želimo imeti to vzdrževanje promptno, v vsakem trenutku. To je zelo občutljiva naloga.

Vzdrževanje telekomunikacij – to je zelo specializirano področje, ki ga običajno ne pokrivamo z lastnimi kadri, pogosto se to opravlja zunaj organizacije.

INFORMACIJSKI SISTEMI IN STANDARDI

- pomen kakovosti na področju IS
- razvoj standardov kakovosti (ISO 9000)
- ISO 9000-3
    - značilnosti standarda (prednosti in slabosti)
- DIN 66285

- značilnosti standarda (prednosti in slabosti)

Govorimo izključno o standardih kakovosti. Tako kot na številnih drugih področjih je vprašanje kakovosti tudi na tem področju IS postalo zelo pomembno že vrsto let nazaj. Uporabniki želijo imeti kakovostne, prijazne in zanesljive rešitve, ki jih je enostavno vzdrževati. In do take rešitve ni tako preprosto priti. Do ugotovitve, da je potrebno nekaj storiti za kakovost inform. rešitev, je prišlo že vrsto let nazaj. V svetu so se začeli razvijati specializirani standardi kakovosti za to področje. Obstaja posebna družina standardov kakovosti in to je ISO 9000. Ta družina je stara že več kot 20 let. ISO standarde razvija mednarodna organizacija za standarde v Ženevi. Pokazalo se je, da ti splošni standardi kakovosti za to področje niso najbolj prikladni, da potrebujemo specializirane standarde. Zato so v 80-tih in 90-tih letih nastali nacionalni standardi in ISO standardi (ISO 9000-3, to je specializiran standard kakovosti za področje informacijskih sistemov in rešitev; drugi za primerjavo, pa je nacionalni nemški standard DIN 66285). Zakaj izpostavljamo ta dva standarda? Ker sta po svoji filozofiji in po svojem namenu ter uporabnosti med seboj povsem različna.

Tako je večina ISO standardov usmerjena v razvojni proces, so v bistvu procesno orientirani standardi, in njihov namen je v zagotavljanju kakovosti proizvodnega procesa, po katerem ta izdelek nastane. To je temeljna filozofija teh standardov. To velja tudi za specializiran ISO standard 9000-3. V bistvu gradi kakovost končnega izdelka na vzpostavljanju celovite kakovosti skozi njegov razvojni cikel. **ISO 9000-3** govori o vzpostavljanju celovitega sistema kakovosti skozi celoten razvojni cikel nekega IS. Ne ukvarja se s kakovostjo končnega izdelka, ampak se ukvarja s samim procesom. Ob predpostavki, če bo razvojni proces kakovosten, potem lahko pričakujemo tudi kakovosten končni izdelek. Razvojni proces IS oz. rešitve opredeljuje skozi posamezne razvojne faze in natančno opredeljuje, kaj moramo storiti skozi posamezne razvojne faze, da bodo izvedene kakovostno. Opredeljuje kakšno dokumentacijo moramo voditi, kdo je zadolžen za vodenje te dokumentacije ipd. Še več, za kakovost tega izdelka sta soodgovorna naročnik in izvajalec. Če želimo, da bo izdelek na koncu kakovosten, potem je potrebno v vse razvojne faze vgraditi preverjanje, kako so bile posamezne aktivnosti izvršene in v to preverjanje mora biti vključen tudi naročnik. To zahteva tudi od naročnika izredno resnost in soodgovornost. Zelo pomembno sporočilo tega standarda je, da je odgovornost za kakovost enakomerno porazdeljena med naročnika in izvajalca, ki se začne že od samega sestavljanja pogodbe, ki natančno opredeljuje vse faze in njihovo preverjanje (podrobnosti o ISO 9000-3 v knjigi). Seveda ima ta standard svoje prednosti in slabosti. To je izrazito procesno usmerjen standard, ki se ukvarja z razvojnim procesom, ne pa s končnim izdelkom. Zelo natančno opredeljuje, kdo je za kaj pristojen, kaj je potrebno storiti, kako morajo biti funkcije med seboj ločene, če hočemo kakovosten rezultat. Pri majhnih organizacijah na področju razvoja to predstavlja velik problem. To v praksi izkazuje …….??????

V nasprotju s tem standardom pa imamo nemški standard **DIN 66285**, katerega strategija je povsem drugačna. Ta standard se pa ne ukvarja z razvojnim procesom, pač pa zgolj s kakovostjo končnega izdelka oz. rešitve. Ta standard skuša opredeliti, kaj so lastnosti, kaj so karakteristike kakovostnega izdelka. Kaj vse mora ta izdelek vsebovati, da ga štejemo za kakovostnega, kako mora biti dokumentiran, katero dokumentacijo mora imeti predloženo, v kakšni obliki mora biti predložena itd.

Glavna razlika med njima je, da se eden ukvarja s kakovostjo procesa, drugi pa s kakovostjo končnega izdelka oz. rešitve. To je zelo pomembna razlika (podrobnosti pa v knjigi…….)