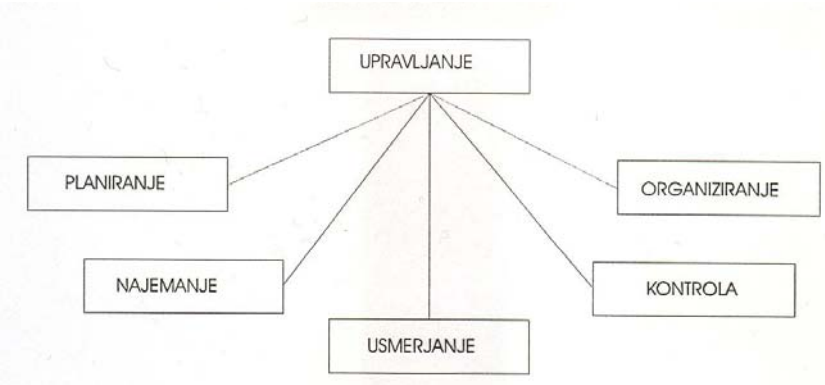
Vodenje projektov informacijskih sistemov

Red. prof. dr. Ivan Rozman

Klasičen model upravljanja 1/2

Slika 1.1



Klasičen model upravljanja 2/2

Tabela 1.1

Aktivnost	Definicija ali razlaga
planiranje	vnaprejšnje določanje smeri akcij za dokončanje organizacijskih ciljev
organiziranje	urejanje in povezovanje dela za dokončanje ciljev in zagotavljanje odgovornosti in pooblastil za pridobitev teh ciljev
najemanje ljudi	selekcija in usposabljanje ljudi za mesta v organizaciji
usmerjanje	ustvarjanje vzdušja, ki bo pomagalo in motiviralo ljudi za dosego želenih rezultatov
kontrola	merjenje in popravljanje učinkovitosti aktivnosti v smeri planiranih ciljev

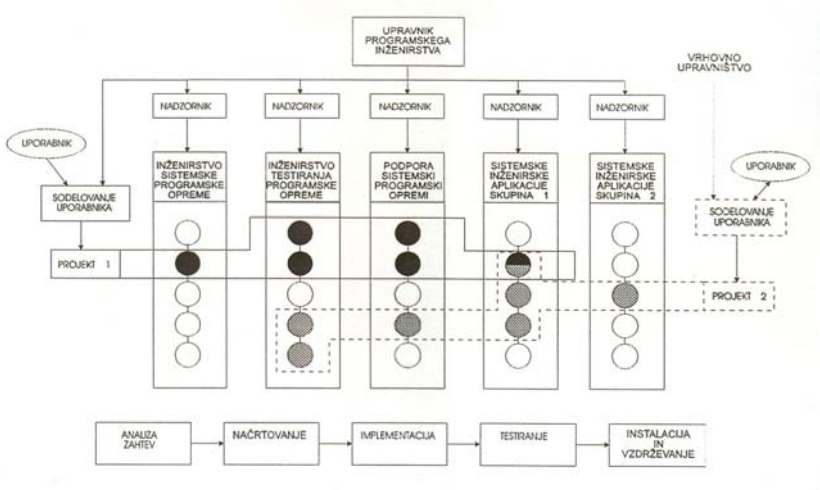
Razporeditev temeljnih upravljalskih aktivnosti po upravljalskih funkcijah 1/2

upravljalske funkcije	temeljne upravljalske aktivnosti
planiranje	- množica ciljev - izdelava strategij - izdelava taktike - ugotovitev smeri akcij - odločitve - množica procedur in pravil - izdelava programov - predvidevanje bodočih situacij - priprava proračuna - dokumentacija projektnega plana
organizacija	- identifikacija in združevanje zahtevanih opravil - izbira in ustanovitev organizacijskih struktur - ustvarjanje organizacijskih mest - definiranje odgovornosti in pooblastil - postavitev pozicijskih kvalifikacij - dokumentacija organizacijskih struktur
najemanje ljudi	- polnjenje organizacijskih mest - asimilacija novo prispelega osebja - izobraževanje in usposabljanje osebja - zagotovitev glavnega razvoja - vrednotenje in ocenitev osebja - kompenziranje - zaključitve določitev - dokumentiranje najemnih odločitev

Razporeditev temeljnih upravljalskih aktivnosti po upravljalskih funkcijah 2/2

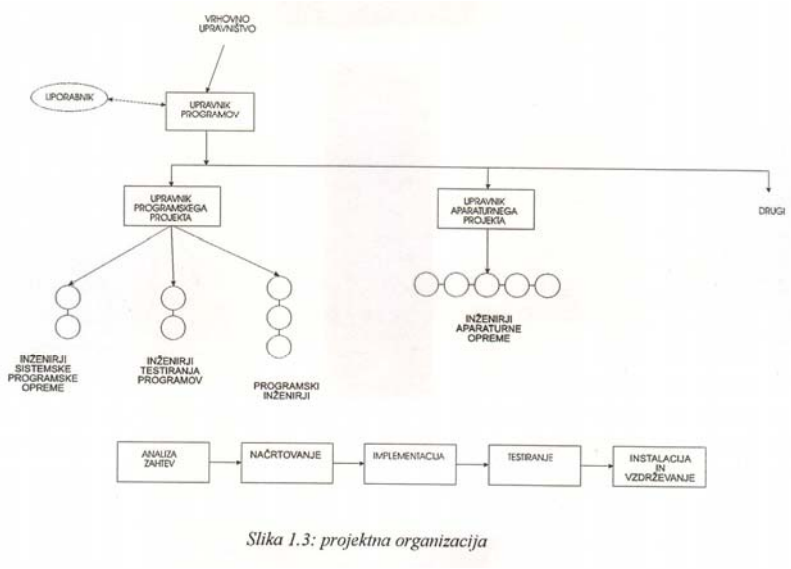
upravljalske funkcije	temeljne upravljalske aktivnosti
usmerjanje	- zagotovitev vodstva - nadzor osebja - zaupanje pooblastil - motivacija osebja - koordiniranje aktivnosti - pospešitev komunikacije - razreševanje konfliktov - upravljanje sprememb - dokumentiranje odločitev usmerjanja
kontrola	- razvoj standardov zmogljivosti - ustanovitev opazovalnih in poročevalnih sistemov - merjenje rezultatov - vpeljava korektivnih akcij - nagrajevanje in disciplina
	- dokumentacija kontrolnih metod

Funkcionalna projektna organizacija



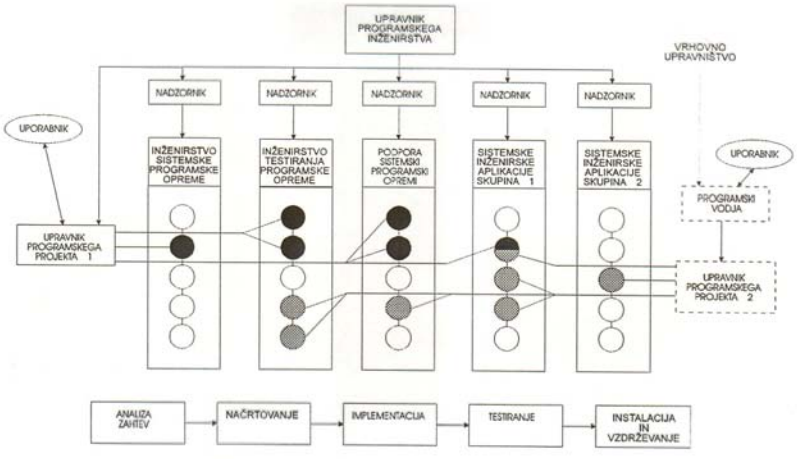
Slika 1.2: funkcionalna projektna organizacija

Projektirana organizacija



Slika 1.3: projektna organizacija

Matrična projektna organizacija



Slika 1.4: matrična projektna organizacija

Prednosti in slabosti različnih tipov projektov programskega inženirstva 1/2

- Funkcionalna projektna organizacija

Prednosti	Slabosti
• organizacija zmeraj obstaja • rekrutiranje, usposabljanje in zadrževanje ljudi je lažje • standardi, tehnike in metode so zmeraj na voljo	• nobena oseba nima popolne odgovornosti in pooblastil nad projektom • probleme vmesnikov je težko rešiti • projekte je težko nadzirati in kontrolirati

- Projektirana organizacija

Prednosti	Slabosti
• obstaja centralna pozicija pooblastil in odgovornosti za projekt • ena oseba ima pooblastila nad vsemi sistemskimi vmesniki • motivacija osebja je običajno visoka	• organizacija se mora formirati • rekrutiranje, usposabljanje in zadrževanje ljudi je težavno • varčnosti ni možno doseči • projekti težijo k temu, da hranijo sami sebe • standardi, tehnike in procedure se morajo izdelati

Prednosti in slabosti različnih tipov projektov programskega inženirstva 2/2

- Matrična projektna organizacija

Prednosti	Slabosti
• izboljšana centralna pozicija odgovornosti in pooblastil nad projektom • vmesnike med funkcijami je mogoče lažje kontrolirati • rekrutiranje, usposabljanje in zadrževanje ljudi je lažje • lažje je začeti in končati projekt (kot pri projektirani organizaciji) • standardi, tehnike in procedure so zmeraj na voljo • boljša in bolj fleksibilna izraba ljudi (kot pri prej naštetih organizacijah)	• odgovornost in pooblastila se delijo med dva ali več upraviteljev (kot projektirana organizacija) • kontrola ali obveznost za izvore (ljudi) je deljena med dva ali vce upraviteljev (kot pri prejšnjih dveh organizacijah) • prelahko premikanje ljudi iz enega na drugi projekt • zahtevano je veliko organizacijske dokumentacije • velika konkurenca pri najemanju izvorov

Prednosti in slabosti različnih tipov projektnih skupin 1/2

- Nesebična projektna skupina

Prednosti	Slabosti
• demokracija dobro deluje v zapletenih in težavnih problemih • izboljšava produktivnosti in kvalitete • zadovoljstvo s službo	• nizka produktivnost zaradi zapletene komunikacije • slabo deluje v naglih razvojih • skupina izbira manj konzervativne in bolj tvegane odločitve, ker nihče zaradi tega ni grajan

- Projektna skupina s šefom

Prednosti	Slabosti
• hiter zaključek opravil • najboljše deluje pri trdih končnih rokih	• zahteva zelo starega in izkušenega šefa programerjev • nizka morala pri projektu • ni primerna za slabo definirane probleme, ki zahtevajo *brainstorming*

Prednosti in slabosti različnih tipov projektnih skupin 2/2

- Hierarhična projektna skupina

Prednosti	Slabosti
• najboljše deluje pri velikih projektih • ena oseba je odgovorna za napore celotne skupine	• slabo deluje pri malih projektih, preprostih opravilih ali raziskovalnih projektih • zahteva veliko nadzora nad razmerjem uslužbencev

Problem upravitelja programskih projektov

Slika 2.1

Problem teorije upravljanja programskih projektov

Slika 2.2

Koraki za ustvarjanje zmagovalne pozicije za vse udeležence v projektu

Tabela 2.1

1. Vzpostavitev množice zmagovalnih predpogojev
   a. razumevanje, kako hočejo ljudje zmagati,
   b. vzpostavitev racionalnih pričakovanj,
   c. usklajevanje človekovih opravil k njihovi zmagovalni situaciji in
   d. zagotavljanje podpornega okolja.
2. Strukturiranje zmagovalnega programskega procesa
   a. vzpostavitev realističnega plana,
   b. uporaba plana za kontrolo projekta,
   c. identifikacija in upravljanje tveganja v primeru situacije zmaga-poraz ali poraz-poraz in
   d. ljudje se naj čutijo vezani s projektom
3. Strukturiranje zmagovalnega programskega produkta
   a. uskladitev produkta z zmagovalnimi situacijami uporabnikov in vzdrževalcev

Izpeljava strateških projektnih smernic za gradnjo zmagovalnih pozicij

Tabela 2.2

predpogoji za situacijo zmaga-zmaga	uporabniki	vzdrževalci	klienti	razvojna skupina
razumevanje zmagovalnih pogojev	analiza poslanstva, operacijski koncept, prototipiranje, specificiran je zahtev, zgodnji uporabniški priročniki	operacijski koncept, operacijske procedure	analiza strošek-korist	potek poti razvoja
realna pričakovanja	piljenje zahtev		dodelitev izvorov	dodelitev izvorov
ujemanje opravil v zmagovalne pogoje	uporabniške specifikacije, pregledi, prototipiranje, vaje	zagotavljanje kvalitete	sledenje stanja, zgodnje odkrivanje napak	najemanje ljudi, organiziranje, zgodnje odkrivanje napak
priprava podpornega okolja	izobraževanje uporabnikov	izobraževanje vzdrževalcev, pretvorba, podporno okolje za dostavo dobrin, konfiguracijsko upravljanje	izobraževanje klientov	izobraževanje razvijalcev, podporno okolje, konfiguracijsko upravljanje

Izpeljava strateških projektnih smernic za gradnjo zmagovalnih pozicij

Tabela 2.3

navodila	uporabniki	vzdrževalci	klienti	razvojna skupina
planiranje procesov	operacijski plan, postavitev in izobraževalni plani	plan za podporo življenjskega cikla	razvojni plani	razvojni plani
kontrola procesov	pregledi	pregledi	sledenje stanja	sledenje stanja, kontrola, povratni efekt zmogljivosti
upravljanje tveganja	uporabniške zahteve, validacija, stabilnost	zagotavljanje kvalitete	proračun, urnik, validacija	proračun, urnik, validacija, najemanje ljudi
vključevanje procesov	sistemski inženirski plan, sodelovanje, pregled sodelovanja, prototipi, vaje	sistemski inženirski plan, sodelovanje, pregled sodelovanja, zagotavljanje kvalitete	pregled strošek-korist, potrditve	pooblastitev, planiranje sodelovanja
strukturiranje produkta	uslužnostna orientacija, učinkovitost, lahko za učenje, lahko za uporabo, prilagodljivo	lahko za spreminjanje, programski standardi	učinkovito, pravilno, izvedljivo	lahko za spreminjanje, uravnoteženo, pravilno

Lista najvišjih deset področij tveganja

Tabela 2.4

PODROČJE TVEGANJA	TEHNIKE UPRAVLJANJA TVEGANJA
1. pomanjkanje osebja	najemanje najboljših talentov, ujemanje z delom, izgradnja delovne skupine, soglasje ključnega osebja, izobraževanje, prerazporejanje ključnih oseb
2. nerealni urnik in proračun	razgradnja predvidevanj več izvornih stroškov in urnika, inkrementalni razvoj, ponovna uporaba programov, piljenje zahtev
3. razvoj napačnih programskih funkcij	analiza organizacije, analiza poslanstva, uporabniška poročila, prototipiranje, zgodnji uporabniški priročniki
4. razvoj napačnega uporabniškega vmesnika	prototipiranje, scenariji, analiza opravil, uporabniške karakteristike (funkcionalnost, stil, delovno opravilo)
5. pozlatitev	piljenje zahtev, prototipiranje, analiza strošek-korist, načrtovanje po stroških
6. ponavljajoče se spremembe zahtev	visoko spremenljv prag, skrivanje informacij, inkrementalni razvoj (odlašanje sprememb na kasnejši postopni razvoj)
7. pomanjkanje zunaj izvedenih komponent	primerjalni testi zmogljivosti, nadzor, preverjanje napotkov, analiza kompatibilnosti
8. pomanjkanje zunaj izvedenih opravil	preverjanje napotkov, pred-dodelitveni pregled, nagradno plačilne pogodbe, konkurenčno načrtovanje ali prototipiranje, izgradnja skupine
9. pomanjkanje zmogljivosti časovno kritičnih komponent	simulacija, primerjalni testi zmogljivosti, modeliranje, prototipiranje, uglasbitev, uglaševanje
10. preseganje zmožnosti računalniške znanosti	tehnična analiza, analiza strošek-korist, prototipiranje, preverjanje napotkov

Produktivnost

Produktivnost merimo kot število linij izvorne kode, ki jo producira programer na mesec. (Enota je človek mesec)

$$produktivnost = \frac{velikost\ \ v\ \ linijah\ \ izvorne\ \ kode}{napor\ \ v\ \ programer\ \ mesecih}$$

Slabosti:

- kaj se razume pod izrazom linija
- več ukazov v eni vrstici
- vpliv programskih jezikov

Prednosti:

- lahko izračunljivo
- omogoča zgodovinsko primerjavo
- vse alternativne metrike imajo probleme

METRIKE IN MODELI

Metrika produkta - zagotovimo lahko avtomatsko zbiranje podatkov

Procesna metrika - ni vezana na program, temveč na proces razvoj informacijskega sistema:

- odvisna je od zunanjih faktorjev
- potrebnih je več meritev in iskanje statistične zakonitosti

METRIKA VELIKOSTI (SIZE METRICS)

Vsem programom, ne glede na jezik je skupno to, da imajo velikost.

Merimo lahko:
- število linij
- število modulov
- število funkcij

Pomembnost teh metrik:
- lahko so izračunljive
- so najpomembnejši faktor večine modelov razvoja informacijskih sistemov
- produktivnost je izražena z velikostjo

DRUGE METRIKE

1. Halstead: operatorji in operandi

2. Albrecht: uporaba funkcijskih točk
   - število vhodov
   - število izhodov
   - število povpraševanj
   - število datotek

3. COCOMO model

HALSTEADOVE METRIKE

n1 = število različnih operatorjev
n2 = število različnih operandov
NI = število vseh operatorjev
N2 = število vseh operandov

- operator je kateri koli simbol ali beseda v programu, ki označuje akcijo
- operand so simboli uporabljeni za predstavitev podatkov:
  - spremenljivke
  - konstante
  - labele

Izpeljava iz Halsteadove metrike

Velikost programa: N=N1+N2
Število programskih vrstic: S_S=N/c [LOC]
c - od programskega jezika odvisna konstanta
Slovar: n=nl+n2

S to množico besed med vsemi besedami, ki jih programski jezik podpira, je možno uspešno napisati program.

Volumen: $V = N \log_2 n$ [bit]
Toliko bitov pomnilnika potrebujemo, da shranimo program, če slovar kodiramo z optimalno dolgimi dvojiškimi besedami

Ocene izpeljane iz Halsteadovih metrik

Ocena dolžine programa: $\hat{N} = n_1 log_2 n_1 + n_2 log_2 n_2$
Programski volumen:V

$V = N \log_2 n$ [bit]

Razlagami si ga lahko kot število potrebnih mentalnih primerjav za izdelavo programa dolžine N in slovarja n.

Potencialni volumen:V^*
To je minimalni volumen s pomočjo, katerega lahko implementiramo nek algoritem.
$V^* = (2 + n_2^*) \log_2(2+n_2^*)$
n_2^* - število konceptualno različnih vhodnih in izhodnih parametrov
2 - zaradi klica procedure in omejilnika, ki nastopa med klicom procedure in listo parametrov
{minimalni volumen za kateri koli algoritem je le klic procedure v kateri je algoritem že kodiran in dodajanje parametrov potrebnih za izvajanje algoritma}
Nivo programa: L
$L=V^*/V$ {L je lahko maksimalno 1- takrat je algoritem zapisan v minimalni velikosti}

Težavnost: D D=I/L

{redundantna uporaba operandov ali neizkoriščanje visokega nivoja kontrolnih struktur povečujejo težavnost}

Ocena nivoja programa: $\hat{L}$

$\hat{L} = \frac{1}{\hat{D}} = \frac{2}{n_1} x \frac{n_2}{N_2}$ formula je definirana zaradi težavnosti določitve V^*

Napor: E

$E = \frac{V}{\hat{L}} = \hat{D}V = \frac{n_1 N_2 N \log_2 n}{2n_2}$ [osnovna mentalna razlikovanja]

Stroud: človeški um je omejen na končno število elementarnih razlikovanj v sekundi $5 \leq ß \leq 20$

Čas: T T=E/ß

predpostavka ß =18 je dajala najboljše rezultate pri Halsteadu

Nivo jezika: λ $\lambda = LV^* = L^2V$ $(\frac{V^*}{V} = L \Rightarrow V^* = LV)$

Obstajajo različni jeziki, ki imajo različno moč. Predpostavlja se, da je λ za nek jezik konstanta {ko V raste se L zmanjšuje}

Izpeljava za E

$$E = \frac{V}{L} = \frac{V}{L}(\frac{L^2}{L^2}) = \frac{\lambda}{L^3}(\frac{\lambda^2}{\lambda^2}) = \frac{V^{*3}}{\lambda^2}$$

dokazali smo, da je trud odvisen od programskega jezika.

UGOTAVLJANJE TRUDA S POMOČJO COCOMO MODELA

Trije nivoji za ugotavljanje truda:
- osnovni (basic)
- srednji (intermediate)
- podrobni (detailed)

Trije načini:
- organski (organic)
- vključen (embedded)
- delno združen (semidetached)

$$E = a_i S^{b_i} m(X)$$

S izvorna koda v 1000 brez komentarja
m(X) kompozitni multiplikator
a in K konstante glede na nivoje in načine

$$E_{nom} = a_i S^{b_i}$$

Ocena časa s pomočjo COCOMO modela
$T = 2.5E^{0.38}$ (organski)
$T = 2.5E^{0.35}$ (vključen)
$T = 2.5E^{0.32}$ (delno združen)

PRODUKTIVNOST IN COCOMO MODEL (Boehm)

atributi produkta:
- RELY - zahtevana zanesljivost programa
- DATA - velikost baze podatkov
- CPLX - zapletenost produkta

atributi računalnika:
- TIME - omejitve izvajalnega časa
- ŠTOR - omejitve glavnega pomnilnika
- VIRT - nestalnost virtualnega stroja
- TURN - čas obrata v računalniku

atributi osebnosti:
- ACAP - zmožnosti analitika
- AEXP - izkušnje s programom
- PCAP - zmožnosti programerja
- VEXP - izkušnje z virtualnim strojem
- LEXP - izkušnje s programskim jezikom

atributi projekta:
- MODP - moderna praksa programiranja
- TOOL - uporaba programskih orodij
- SCED - uporaba razvojnega urnika

Prednosti COCOMO modela pred WF modelom:
- manj atributov, lažje upravljanje
- večja neodvisnost atributov
- večja gradacija in natančnost

Ocena COCOMO modela

1. m(X) [0.088,72] => ocena velikosti 800:1
2. preveč parametrov
3. empirično določene konstante

Modifikacije COCOMO modela

kalibracija:

$E = 2.6S^{1.08}m(X)$ (organski)

$E = 2.9S^{1.12}m(X)$ (vključen)

$E = 2.9S^{1.20}m(X)$ (delno združen)

VERIFIKACIJA IN VALIDACIJA V&V

Verifikacija je ugotavljanje, ali sistem gradimo pravilno

Validacija je ugotavljanje, ali je sistem pravilno zgrajen

SPLOŠNI MODEL V&V PROCESA

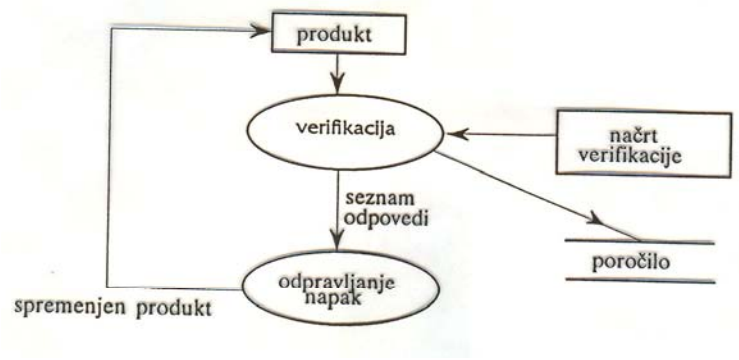
Verifikacijski sistem tvorijo postopki, orodja in preverjalci. **V&V proces** je izvajanje postopkov verifikacijskega sistema.

KAZALCI V&V PROCESA

- Preverjanje je proces.
- Stanje procesa se lahko spremeni ob določenem dogodku (korak)
- Preverjanje je časovno diskreten sistem.
- Preverjanje je drago
- Preverjanje končamo na temelju *objektivnih kvantitativnih kazalcev trenutnega in zahtevanega stanja V&V procesa ali produkta,*
- Kazalci za vrednotenje stanja V&V procesa so v bistvu metrike, ki jih izberemo toliko, da lahko stanje procesa dovolj natančno opišemo.

MODEL V&V PROCESA

Časovno nespremenljiv model:	napake samo beležimo,
Časovno spremenljiv model:	napake takoj odpravljamo (redko)



Najpomembnejši napotki za V&V

1. Verifikacije in validacije ne izvajajmo brez predhodnega V&V načrta.
2. V&V proces ni imun na napake.
3. V&V procesa ne smemo podcenjevati.
4. Pazljivo moramo pregledati dejansko stanje.
5. Število odkritih napak veča verjetnost obstoja Se neodkritih.
6. Tudi pri odpravljanju napak lahko vnesemo nove.
7. Objekt verifikacije se med preverjanjem ne sme spreminjati.
8. Praktično je nemogoče odkriti vse napake.
9. Globalni cilj V&V procesa ni odkrivanje prisotnosti napak, ampak zmanjšanje stroškov razvoja in vzdrževanja ter dvig kakovosti produkta.
10. Vsaj za eno odpoved mora biti določena škoda (stroški), ki nastanejo pri njenem nastopu.
11. Lastnosti produktov, ki jih ni mogoče preverjati, je tudi nesmiselno realizirati.
12. Dosledno dokumentiramo V&V proces.

NERAČUNALNIŠKE METODE V&V

Stroški odpravljanja napak strmo naraščajo glede na oddaljenost od začetne točke faze projekta.

Uporabljamo razne oblike preverjanja, ki temeljijo na psiholoških metodah.

Značilnosti:

- so izjemno uspešne,
- dvignejo kakovost produkta in procesa,
- ne zahtevajo večjih stroškov in
- uporabne so v celotnem življenjskem ciklu.

Poznamo različne metode neračunalniške V&V:

- pregledi
- inšpekcije in
- presoja (audit).

Člani okolja, ki izvajajo neračunalniške V&V morajo poznati odgovore na naslednja vprašanja:

- kakšen je namen verifikacije,
- kdaj se izvaja,
- kaj bo pregledano in
- kakšni so ocenjevalni kriteriji.

PREGLEDI

So posebni verifikacijski postopki, ki se nanašajo na projekt ali pa na produkte posameznih faz.

Preglede oz. formalne sestanke delimo na:

- splošne tehniške preglede,
- inšpekcije in
- sprehode.

INŠPEKCIJA (M. E. FAGAN 1976)

Poudarek je na identifikaciji in odpravljanju napak. Najpogosteje preverjamo izvorno kodo.

Pogoji za inšpekcijo:

- Določeni morajo biti sprejemni kriteriji, katerim mora zadostiti objekt preverjanja, da se ga sprejme v postopek inšpekcije.
- objekt mora biti vidljiv,
- določena mora biti terminalna kriterijska funkcija in
- določena mora biti odpovedna kriterijska funkcija.

Postopek

1. planiranje,
2. seznanjanje s produktom,
3. priprava na sestanek,
4. sestanek,
5. odprava pomanjkljivosti oziroma napak in
6. kontrola odprave pomanjkljivosti

Najdene napake razvrstimo glede na izbrane atribute (vrsta, pomembnost napake)

VALIDACIJA

Validacija je proces preverjanja, pri katerem ugotavljamo, ali je izdelek pravilno grajen

1. Vhodni dokumenti (kompleten program z dokumentacijo, uporabnikove zahteve, načrt validacije).
2. Metode (testiranje).
3. Posebnosti (napak ne popravljamo).
4. Izhodni dokumenti (končno V&V poročilo).

V okviru validacije izvajamo:
- funkcijski test,
- performančni test,
- testiranje stresa,
- strukturni stres in
- druge vrste verifikacije.

prekoračitveno testiranje, testiranje zaščite, testiranje pomnilnika, verifikacija konfiguracije, verifikacija dokumentacije, verifikacija kompatibilnosti, verifikacija varnosti, verifikacija prijaznosti in razumljivosti.

V&V PRED RAZVOJNO FAZO

Pred razvojno fazo preverjamo:
- uporabnikove zahteve,
- smotrnost nakupa / izdelave in
- sistemske specifikacije.

Verifikacija uporabnikovih zahtev
1. Vhodni dokumenti (zahteve v pisni obliki, ki jih je podpisal naročnik)
2. Metoda (neračunalniška verifikacija)
3. Posebnosti (obvezno sodelovanje naročnika)
4. Izhodni dokumenti (poročilo verifikacije, osnutek načrta validacije, osnutek načrta za prevzemni test)

Identifikacija globalnih zahtev
- funkcijske zahteve,
- performančne zahteve,
- zahteve glede aparaturne opreme,
- zahteve glede na dialog uporabnik - stroj,
- zahteve glede na določene standarde in
- zahteve glede stroškov in rokov.

Preverjanje zahtev
- naročnik,
- neodvisna preverjalna organizacija skupaj z naročnikom ali
- izdelovalec skupaj z naročnikom.

V&V V RAZVOJNI FAZI

Ta faza je razdeljena v 3 podfaze:

Podfaza	objekt verifikacije
načrtovanje implementacije	dokumentacija strukturnega načrtovanja
kodiranje modulov	moduli
integracija	integriran program

Verifikacija dokumentacije strukturnega načrtovanja

1. Vhodni dokumenti (dokumentacija strukturnega načrtovanja, standardi, sistemske specifikacije),
2. Metode (neračunalniške verifikacijske metode, analiza sledljivosti, verifikacija strukturnega načrtovanja),
3. Izhodni dokumenti (poročilo o opravljenem V&V procesu, načrt testiranja modulov).

Verifikacija modulov in integracije

1. Vhodni dokumenti (uporabnikove zahteve, sistemske specifikacije, dokumentacija strukturnega načrtovanja),
2. Metode (neračunalniška in računalniška verifikacija, V&V znotraj orodij ČASE)
3. Posebnosti (poudarek je na testiranju)
4. Izhodni dokumenti (poročilo o opravljenem V&V procesu, načrt validacije)

Kritični faktorji uspeha

- enostavnost uporabe,
- vzdrževanje programske opreme,
- stroški,
- prenosljivost,
- zanesljivost,
- varnost in
- reakcijski čas na vitalne signale.

Osnutek validacijskega načrta

Le redko lahko ocenimo vse kritične faktorje uspeha v tej fazi, zato jih preverjamo šele pri validaciji.

Preverjanje zahtev

Opravimo jih lahko z eno izmed neračunalniških verifikacij

Izvedljivost zahtev

Neizvedljivost zaradi:

- fizične omejitve,
- prevelikih stroškov,
- pomanjkanja časa in
- nesposobnosti izdelovalca.

Odprava pomanjkljivosti oziroma napak

- avtor ali skupina odpravi napake
- skupina za V&V pregleda seznam nepravilnosti oz. napak in če je treba dopolni kontrolno listo.

Kontrola odprave pomanjkljivosti

- ponovi se inšpekcija, da se preveri resnično odpravo napak.

TRAJANJE IN UČINKOVITOST INŠPEKCIJE

Trajanje:

- maksimalno 2 polni uri,
- priprava traja dalj časa kot sestanek sam,
- za uspešnost uporabljamo preproste metrike
  - (izvorna koda v tisočih ali specifikacije ali zahteve v vrsticah ali straneh)

Učinkovitost:

- 7-20 pomembnih napak na 1000 vrstic izvorne kode brez komentarjev,
- za vsako pomembno napako je potrebnih 1-5 ur,
- inšpekcija je 2-10 krat uspešnejša do testiranja,
- ne more nadomestiti testiranja,
- najdemo napake, kijih s testiranjem ne bi našli,
- stroške v dnevih izračunamo: ***št. dni = 3*kLOC***
  zapravimo 15-25% celotnega časa projekta
- vprašanje ekonomske učinkovitosti.

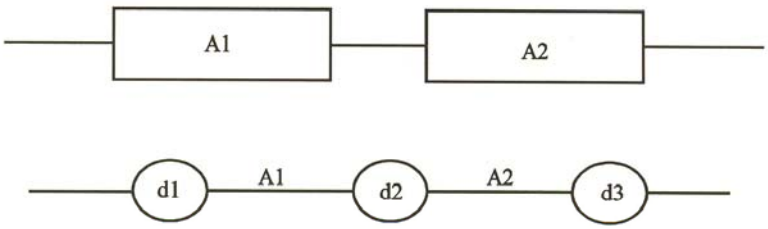
Navodila za pripravo urnika 1/2

- Ne delamo razporeditve na bazi 8-urnega delovnika. Zaradi dejavnosti, ki se ne tičejo projekta in nepredvidljivosti, je bolje, da planiramo delovni dan z manj urami. Med nepredvidljivosti spada bolezen, zamenjave osebja, posebne zadolžitve, izgube časa, izmenjavanje informacij ...
- Idealno je, če za neko opravilo zadolžimo eno osebo, razen če je opravilo očitno primerno za več ljudi. Če katero opravilo, ki ni očitno primerna za več ljudi zahteva več ljudi, ga razstavimo na enostavna opravila, ki jih lahko opravi en človek.
- Ne razporejajmo posameznika na preveč prekrivajočih se opravil. Neizkušen posameznik naj dela samo eno stvar v določenem času. Izkušeni posamezniki lahko izvajajo več opravil hkrati, vendar vseeno pazimo, da jih ne obremenimo z več kot 4 aktivnostmi.
- Dovolimo praznike, počitnice in izpopolnjevanje (to se največkrat pomotoma pozabi). Pripravimo urnik za vsakega posameznika, da se izognemo tem pastem.

Navodila za pripravo urnika 2/2

- Povečujemo oz. zmanjšujemo število delovnih dni glede na različno stopnjo usposobljenosti.
- Dodeljujemo dovolj časa za komunikacijo med člani skupine in to uvrščamo v plan. Več kot sodeluje ljudi pri projektu, več časa je treba posvetiti tej komunikaciji.
- Pripravimo si mejnike (milestone) v različnih fazah za povzetek informacij o projektu. Veliko projektov bo že po naravi imelo nekaj teh točk, ki bodo stičišče ključnih aktivnosti in delujejo kot testne točke za člane tima, vodstveni kader in uporabnike.

Aktivnostno in dogodkovno orientiran diagram

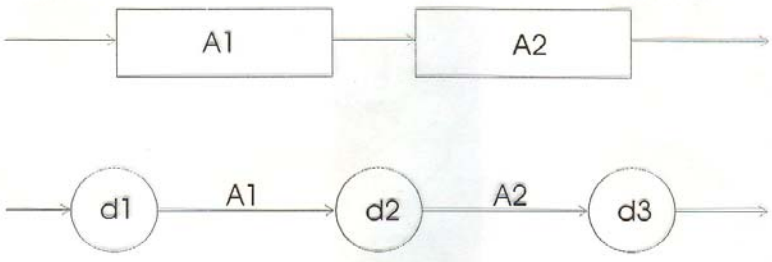


Kritična pot je najdaljše trajanje niza aktivnosti.

Pri sestavljanju mrežnih diagramov je potrebno paziti na naslednje:
- katere aktivnosti morajo biti končane pred pričetkom naslednje aktivnosti
- katere aktivnosti lahko potekajo sočasno
- katere aktivnosti se morajo pričeti po koncu določene aktivnosti
- katere aktivnosti morajo biti končane istočasno

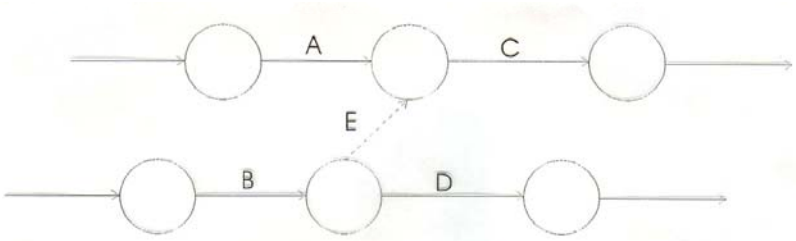
Pravila pri izdelavi mrežnih diagramov:
- zaporedje aktivnosti poteka iz leve proti desni
  diagram naj bo preprost, brez prepletanj (vodoravne povezave)
- vsaka aktivnost se mora pričeti in končati z dogodkom
- če se neka aktivnost ne more pričeti pred zaključkom določene aktivnosti, potem naj imata aktivnosti skupen dogodek, ki predstavlja konec ene in pričetek naslednje aktivnosti
- če konec več aktivnosti predstavlja pogoj za začetek naslednje aktivnosti se morajo predhodne aktivnosti končati v začetnem dogodku naslednje aktivnosti
- če je konec predhodne aktivnosti pogoj za pričetek več naslednjih aktivnosti potem postane končni dogodek predhodne aktivnosti začetni dogodek vseh naslednjih aktivnosti
- če imata dve ali več aktivnosti skupen začetni in končni dogodek to ponazorimo z uvajanjem fiktivnih aktivnosti na začetnem ali končnem dogodku ene izmed aktivnosti

Aktivnostno in dogodkovno orientiran diagram

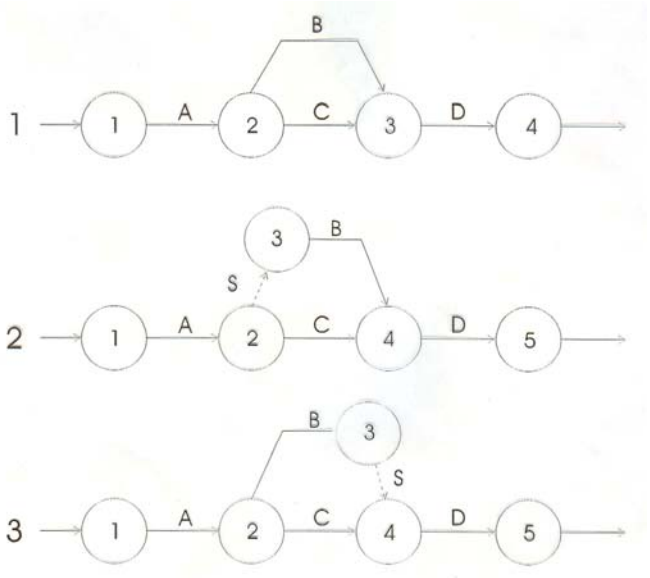


Slika 3.1: aktivnostno in dogodkovno orientiran diagram

Fiktivna odvisnost

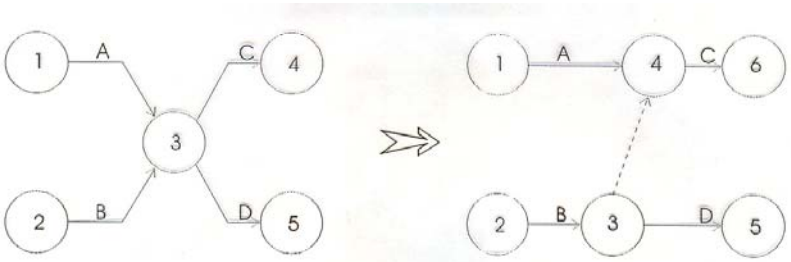


Slika 3.2: fiktivna aktivnost



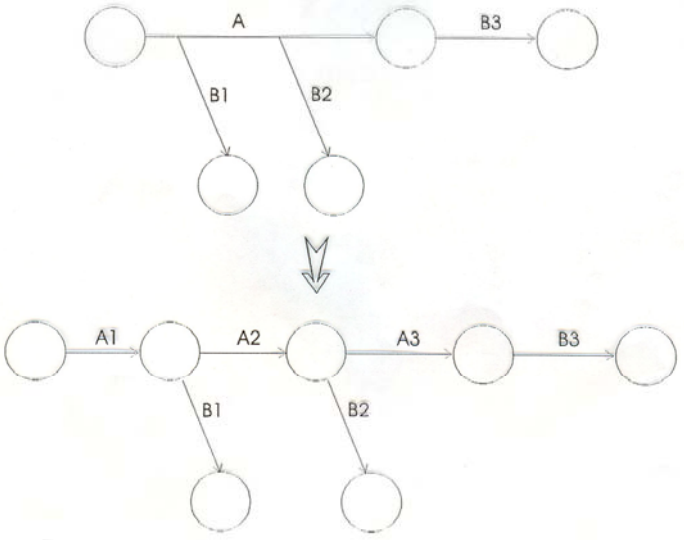
slika 3.3: uvajanje fiktivne aktivnosti za enoznačno identifikacijo poti

Sprememba grafa v enoznačno obliko 1/2



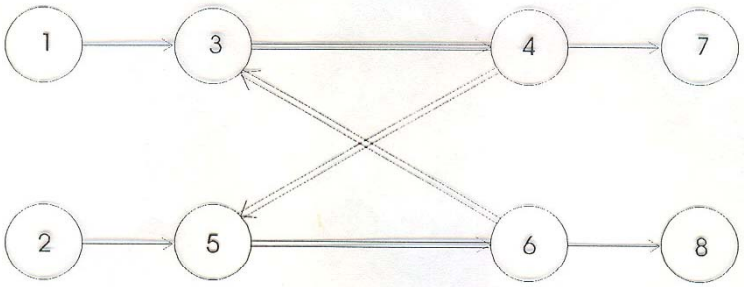
slika 3.4: sprememba grafa v enoznačno obliko

Sprememba grafa v enoznačno obliko 2/2



slika 3.5: sprememba grafa v enoznačno obliko

Primer nepravilnega grafa



slika 3.6: nepravilen graf

Številčenje mrežnega diagrama 1/2

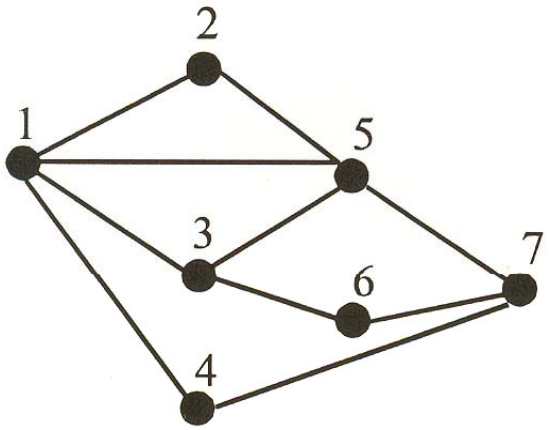
Vse dogodke mrežnega diagrama je potrebno oštevilčiti. Obstajata dva načina številčenja:

1. **poljubno številčenje:** vsakemu dogodku se določi poljubna številka. Edina omejitev je, da mora vsak dogodek imeti svojo številko. Tak način ima naslednje pomanjkljivosti:
   - težave v identifikaciji posameznih aktivnosti in pogoste napake v vrstnem redu pisanja številk (npr. 10-16 namesto 16-10)
   - uporaba računskih postopkov za analizo mrežnega diagrama je otežena, saj ni ključa za vrstni red obdelave podatkov
   - večina pripravljenih rutin za uporabo računalnikov zahteva rastoče številčenje
   - težko je odkriti zaprte zanke

2. **rastoče številčenje:** dogodke oštevilčimo od 1 do n pri tem ima začetni dogodek številko 1 končni pa n. Številčenje je lažje z uporabo Fulkersonovega pravila.
   - prvi dogodek oštevilčimo z 1
   - aktivnosti, ki iz njega izhajajo prekrižamo
   - prekrižamo vse aktivnosti, ki izhajajo iz posameznega dogodka začevši pri dogodku, ki je v diagramu najvišje in ima prekrižane vse predhodnje aktivnosti

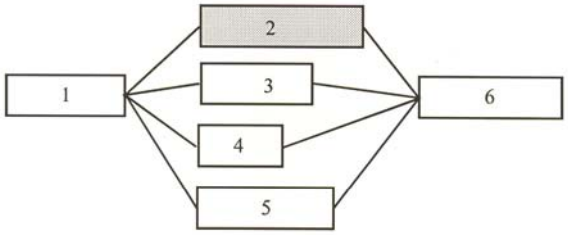
Številčenje mrežnega diagrama 2/2

Primer:



Kritična aktivnost in kritična pot

- kritična aktivnost je tista aktivnost, ki v nekem delu procesa traja najdalj
- kritična pot je pot, ki vodi preko kritične aktivnosti



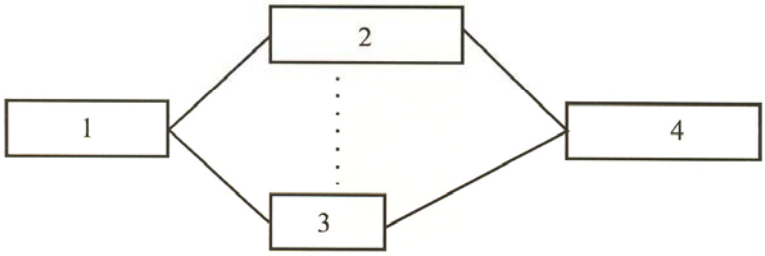
druga aktivnost v primeru je kritična

kritična pot je 1-2-6

Časovna analiza

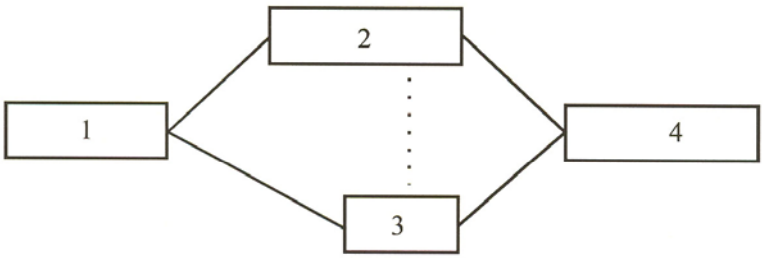
S časovno analizo določamo trajanje aktivnosti, ki so v mrežnem diagramu. Precizno določanje časa trajanja posamezne aktivnosti je pogojeno s točnim opisom predvidenih postopkov za njeno izvršitev. Pri tem je potrebno upoštevati vrsto in število delavcev, število strojev in drugih pomožnih sredstev, kot tudi način njihovega dela.

DOLOČANJE NAJHITREJŠEGA ZAČETKA IN NAJHITREJŠEGA KONCA AKTIVNOSTI

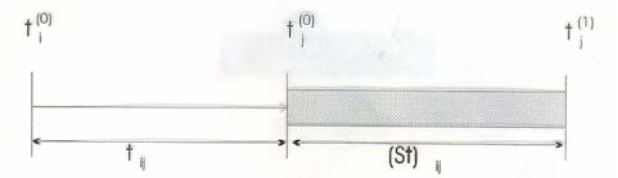


DOLOČANJE NAJKASNEJŠEGA ZAČETKA IN NAJKASNEJŠEGA KONCA AKTIVNOSTI

- poteka podobno kot določanje najhitrejšega začetka in konca le da začnemo pri končnem dogodku projekta in se pomikamo proti začetnemu.
- z določanjem lahko začnemo šele, ko smo določili najhitrejši zaključek za vse aktivnosti

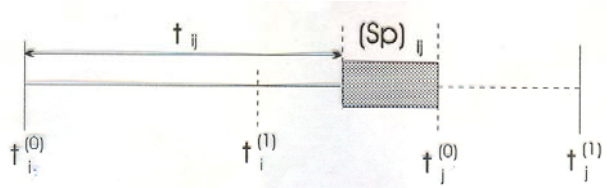


Skupna časovna rezerva



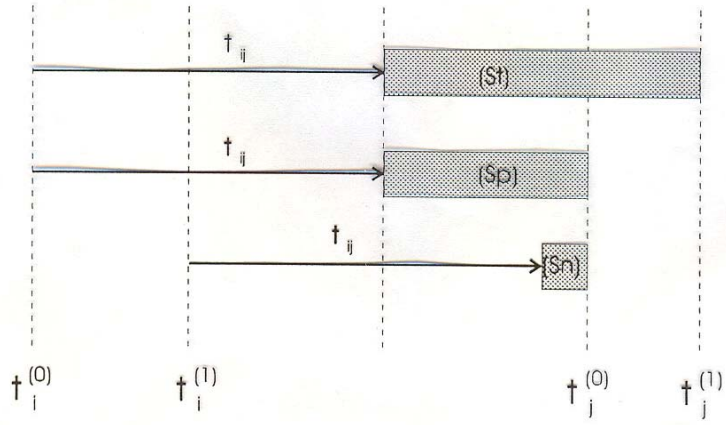
slika 3.7: skupna časovna rezerva (St)

Prosta časovna rezerva



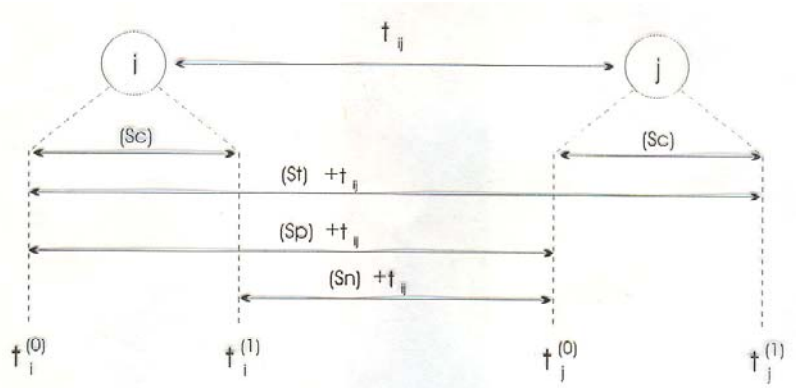
slika 3.8: prosta časovna rezerva (Sp)

Trije tipi časovnih rezerv



slika 3.9: trije tipi časovnih rezerv

Vsi tipi časovnih rezerv



slika 3.10: vsi tipi časovnih rezerv