Vodenje projektov in osebne produktivnosti

Predloge za predavanja predmetov:
Vodenje projektov
Projektni praktikum

Franc Solina

5. januar 2011

Kazalo

Poglavje 1

Uvod

- hitre spremembe, v poslovnem in zasebnem življenju,
- tradicionalna hierarhična organizacija izginja zaradi enostavnega razširjanja in dostopa do informacij
- poplava informacij, vedno več možnosti se odpira posamezniku
- prepletanje zasebnega in poslovnega življenja
- vedno manj časa!?!
- potrebna je dobra organizacija

Poglavje 2

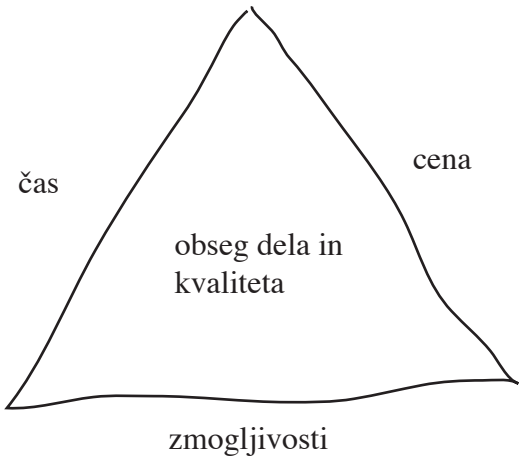
Kaj je projekt?

- zaporedje aktivnosti, aktivnost je zaokrožena delovna celota
- enkratne, kompleksne, soodvisne aktivnosti
- en cilj
- časovni okvir
- v okviru proračuna
- upoštevajoč specifikacije

2.1 Parametri projekta in kako vplivajo na obseg projekta

- obseg projekta: kaj je potrebno narediti, kaj naj bi produkt delal?
- kvaliteta produkta, kvaliteta procesa
- TRIJE parametri:
  1. cena
  2. čas
  3. zmogljivosti: ljudje, oprema, prostori,

Med izvajanjem pa hitro pride do sprememb:



Slika 2.1: Trikotnik soodvisnosti. Ko je načrt projekta narejen, je trikotnik uravnotežen. Sprememba katerekoli stranice vpliva na ploščino!

- sprememba obsega
- zamude pri doseganju podciljev
- delo ni učinkovito
- dodajanje funkcionalnosti, ki niso v zahtevah

Vrste projektov:

- tveganje
- poslovna vrednost
- trajanje
- kompleksnost
- uporabljena tehnologija
- cena

Od tveganih, dragih in pomembnih projektov do kratkih, enostavnih in rutinskih

Zaradi hitrih sprememb (zahteve, tehnologija, spremebe okolja) so spremembe projekta med izvajanjem postale vedno bolj pogoste. Cilje oziroma zahteve je vedno težje natančno definirati.

Tradicionalne metode projektnega vodenja so primerne za dobro definirane cilje, v stabilnem okolju (razvite za inženirje, gradbeno in strojno industrijo).

Pri razvoju programske opreme so se pojavile nove ekstremne metode vodenja projektov (agilne metode, ekstremno programiranje, SCRUM, Crystal, itd.).

Vmes pa so adaptivne metode, ki nudijo okvir vendar ne zahtevajo vnaprej natančnega planiranja.

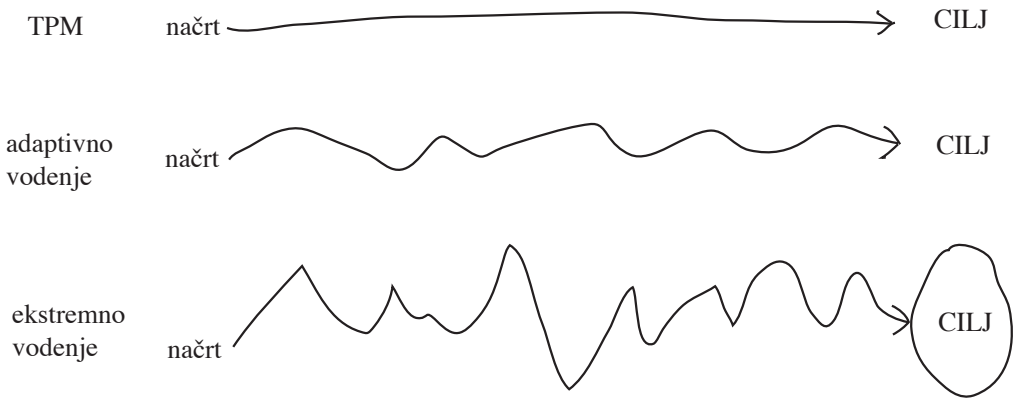
Tabela 2.1: Tri metodologije vodenja projektov

		pot do cilja: def.	pot do cilja: ni def.
cilj	def.	TPM	adaptivno
	ni def.	–	ekstremno

TPM (Tradicionalni projektni management) — zahteva podroben nacrt preden se projekt začne

adaptivno vodenje — zahteva tudi podroben načrt, vendar ni v vseh podrobnostih narejen pred začetkom dela, ampak ko je to potrebno

ekstremno vodenje — ni načrta kot pri TPM in adaptivnem, ugibanje in poskušanje



Slika 2.2: Tri vrste projektov: tradicionalni projektni management, adaptivno in ekstremno vodenje.

Poglavje 3

Tradicionalni projektni management

Ko lahko dokaj zanesljivo definiramo cilje projekta in na tej osnovi načrtujemo projekt.

Sicer uporabimo adaptivno vodenje projektov.

Slika 3.1: Faze projekta

1. Definiranje projekta

   - Kaj je problem ali priložnost?
   - Kaj je cilj projekta?
   - Kaj moramo narediti, da dosežemo cilj?
   - Kako bomo ugotovili, ali je projekt uspešen?
   - Kaj so predpostavke, tveganja ali ovire za uspeh projekta?

   v času projekta pride do sprememb

2. Načrtovanje

   planiranje izguba časa, saj vedno pride do sprememb

   planiranje ni pravo delo, planiranje je naporno!

   ni casa za planiranje, je pa cas za popravljanje slabo opravljenega dela

   koristi planiranja:

   (a) planiranje zmanša negotovost
   planiranje zahteva, da premislimo o moznih potekih projekta

   (b) planiranje poveča razumevanje
   sam akt planiranja zahteva globji premislek o ciljih in poteku dela

   (c) planiranje poveča učinkovitost
   nekatere aktivnosti lahko izvajamo paralelno, resurse uporabimo bolj učinkovito itd.

3. Izvajanje vsi člani projektne skupine začnejo izvajati aktivnosti po načrtu, saj mora vsak vedeti, kaj in kdaj mora kaj narediti

4. Kontroliranje Ne glede na to, kako skrbno smo planirali, pride do sprememb.

   Vodja projekta mora poskrbeti, da kontrolira izvajanje projekta in predlaga spremembe, če so potrebne.

5. Zaključevanje

   Formalni konec projekta, predaja rezultatov naročniku, dokumentiranje projekta za lažje kasnejše delo, analiza projekta (koliko je nacrt uresnicen, kaj smo se naucili?)

3.1 Tri variante TPM

1. Definiranje, planiranje, organiziranje

   ni kontrole, obicajno dela le en clovek, gre za boljso organizacijo dela, dolocitev casovnega okvira, priti do seznama kaj je potrebno narediti

2. Definiranje, planiranje, izvajanje, kontroliranje

   zaceti projekt je pol dela, vec kot je ljudi, aktivnosti, virov, vec kontrole je potrebno

   potreben je mehanizem, ki cimprej odkrije probleme in jih pomaga resiti

3. Definiranje, planiranje, izvajanje, kontroliranje, zakljucevanje

   dober vodja projektov se stalno uci na osnovi analize preteklih projektov

3.2 Vodenje kvalitete

- stalno izboljsevanje kvalitete
- kvaliteta produkta, storitve, procesa
- kvaliteta vodenja projekta samega

3.3 Upravljanje s tveganji

ocena tveganja, da se nekaj zgodi v prihodnosti:

1. verjetnost, da se nekaj zgodi
2. višina škode, če se to zgodi
3. produkt verjetnosti in potencialne škode, ali je nad pragom?

Identifikacija tveganj
Ocenjevanje tveganj
Nacrtovanje odgovora, ce do dogodka pride
npr. kaj ce oddide glavni programer?
Spremljanje tveganj in kontrola

3.4 Naročanje resursov

za izvedbo projekta je pogosto potrebno kupiti, najeti, zaposliti, izbrati opremo, ljudi, izvajalce

Načrtovanje naročanja: dilema kupiti/narediti

Izdelava Razpisa za izvajalce, dobavitelje (javno naročanje itd.)

Izbira dobaviteljev oziroma izvajalcev

Nadzor izvajalcev

zaključevanje pogodbe (ali je projekt končan?, ali je bilo vse narejeno?, arhiviranje vseh dokumentov)

3.5 Komuniciranje

zakaj je dobro komuniciranje med vsem deležniki v projektu pomembno?

načini komuniciranja: sestanki, poročila, sodobna informacijska tehnologija (email, video konference)

```
dobro planiranje: hribcek na zacetku

slabo planiranje: krivulja stalno narasca
```

SLIKA: bolecina, dobro planiranje in slabo planiranje (pay now or pay later)

Poglavje 4

Definiranje projekta

- Definiranje je 1. faza v poteku projekta
- izvor številnih problemov so nejasno definirani cilji ali nesporazumi med naročnikom in izvajalcem:
- COS Conditions of satisfaction: naročnik ga mora potrditi
- dokument za boljšo komunikacijo: POS - Project overview statement: dinamičen dokument, ki se stalno dopolnjuje, ko so znane nove informacije!
- POS: vzpostavitveni dokument projekta, projektna naloga, projektna listina
- Zakaj je definiranje ciljev težavno: disproporc med željami in dejanskimi potrebami pri naročniku: NEEDS/WANTS
- pogovarjanje in poslušanje!!! kot osnovni način analize

SLIKA: elipsa: zahteva, razjasni zahtevo, odgovor, uskladi odgovor
COS $->$ POS
POS (na eni strani), za nadrejene, za projektni tim, za narocnika, ...
POS vsebuje naslednje elemente:

1. Problem ali priložnost (znani problemi, zahteva naročnika, iniciativa zaposlenih ...)
2. cilj projekta (SMART: Specific, Measurable, Assignable, Realistic, Time-related)
3. kaj je potrebno narediti (bolj podrobno razdelan cilj, podcilji) ponovno razmisli o mejah projekta,

4. kriterij za uspeh: povečan dobiček, zmanjšani stroški, boljša storitev
5. predpostavke, tveganja, ovire: (tehnološke, okoljske, medosebne, kulturne, spremembe okolja)

Priloge POS: analiza tveganj, finančne analize (analiza izvedljivosti, analiza stroškov/prednosti, analiza pokritja stroškov, povračila investicije)

4.1 Proces odobritve projekta

Kdo sodeluje pri odobravanju?

- jedro projektne skupine
- projektna skupina
- vodja projekta
- vodje resursov in funkcijskih enot
- narocnik
- nadrejeni

Kriteriji za odobritev?
POS $->$ PDS: Project definition statement
bolj podrobno razdelan dokument namenjen predvsem projektni skupini

Poglavje 5

Identifikacija projektnih aktivnosti

2. faza v projektu je Načrtovanje, ki je sestavljeno iz:

- identifikacije aktivnosti
- ocene trajanja, virov in stroškov aktivnosti
- izdelave in analiza mrežnega diagrama

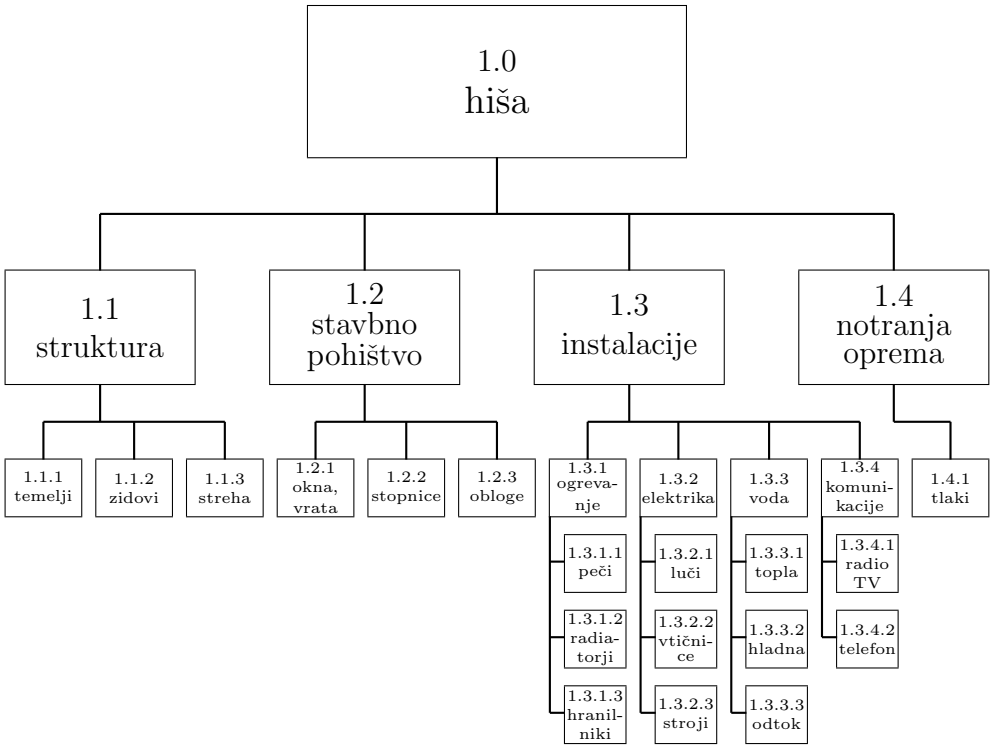
5.1 WBS

hierarhicna, funkcijska, objektna dekompozicija na aktivnosti
Uporaba WBS:

- proces za razmisljanje
- orodje za nacrtovanje arhitekture
- orodje za planiranje
- orodje za porocanje

5.2 Kako pridemo do WBS?

zaradi velikega stevila ljudi, velik papir, post-it listiki na tabli
od zgoraj navzdol,
od spodaj navzgor
mindmapping (graficna metoda, Tony Buzan)



Slika 5.1: Delovna struktura—WBS (Work Breakdown Structure) za gradnjo hiše

5.3 Dobro definirana aktivnost

1. Status, dokončanje aktivnosti je merljivo
2. Zacetek in konec aktivnosti jasno definiran
3. Aktivnost ima rezultat
4. cas in stroske aktivnosti je mozno oceniti
5. dolzina aktivnosti je v razumnih okvirih
6. delo na aktivnosti je neodvisno

5.4 Kako zgradimo WBS?

1. Samostalniski (noun-type):
   - delitev po fizičnih komponentah (npr. hiša)
   - ali delitev po funkcijskih komponentah (avto: pogonski del, šasija)
2. Z glagoli (verb type):
   - ponavadi v povezavi z neko metodologijo, npr. design-build-test-implement itd.
3. Organizacijski pristop: geografski, oddelčni

Poglavje 6

Ocena trajanja, virov in stroškov aktivnosti

2. del v fazi Načrtovanja projekta.
Ocenja trajanja, problematika:

- trajanje aktivnost in potrebno delo ni ISTO: betoniranje plošče, ne moremo nadaljevati, dokler se ne posuši!
- trajanje vkljucuje le delovne dni, brez vikendov in praznikov
- cas fokusiranega in neprekinjenega dela in delo s prekinitvami
- realno: 75% koncentriranega dela in 33% neplanirane prekinitve (telefon, obiski, e-mail itd.) $->$ 50% izkoristek casa (10 urno delo potrebuje 20 ur delovnega casa)

6.1 Razmerje med trajanjem aktivnosti in resursi za aktivnost

obicajno ni linearno razmerje med tranjanjem aktivnosti in resursi za to aktivnosti, npr. prenaănje stolov iz enega prostora v drug prostor!
delitveno in kompleksno delo
težka delitev kompleksnega dela, nakar je potrebno rezultate še integrirati
SLIKA: trije grafi, resursi na X osi, čas na Y osi,
a) krivulja konkavno pada,
b) konstantna - horizontalna, ker delo ni deljivo

c) konkavna, ki zopet narasca, ker več ljudi delo še podaljša

6.2 Variacija v trajanju aktivnosti

različno znanje/izkušnje
nepredvideni dogodki
učinkovitost dela, prekinitve
napake in nesporazumi

6.3 Šest metod za ocenjevanje trajanja aktivnosti

Splošne metode ocenjevanja:

1. podobnost z drugimi aktivnostmi
2. zgodovinski podatki
3. nasvet ekspertov
4. delfi tehnika (več ocenjevalcev, več krogov, ko ocenjevalci lahko modificirajo svoje ocene)
5. tri tockovna tehnika (optimisticna, pesimistcna, najbolj verjetna ocena trajanja)
6. wide band Delphi technique: delfi + tri tockovna (ocenjevalci povedo tri ocene!)

Na posameznih področjih (gradbeništvo, razvoj programske opreme itd.) je cela kopica specialnih metod za ocenjevanje!

Zavedaj se, da če bi isto aktivnost večkrat ponovili, bi trajanje skoraj vedno bilo različno! Zakaj?

Kako natancne so ocene?

6.4 Ocena virov

- ljudje (najtežje ocenjevati, pri intelektualnem delu velike razlike v produktivnosti!)
- prostori (pisarne, proizvodni prostori)

- oprema (računalniki, programska oprema itd.)
- denar
- surovine, sestavni deli

Ljudje imajo različna znanja in kvalifikacije!
Kategorije kvalifikacij in nivoji kvalifikacij!
Porazdelitev ljudi po aktivnosti s pomočjo matrike kvalifikacij:

```
        kvalifikacije       kvalifikacije             aktivnosti
        ----------            ----------                ------------
aktivn. |          +  Ljudje  |             =  Ljudje  |

   inventorij potreb    inventorij kvalifikacij
```

Podoben digram kot za WBS lahko naredimo tudi za resurse.
Ocenjevanje trajanja aktivnosti je odvisno od:

- čas trajanja aktivnosti
- kolikšen delež delovnega časa bo človek lahko posvetil tej aktivnosti
- koliko dela lahko človek v celoti porabi za za aktivnosti?

Načrt resursov:

- več ljudi nujno ne skrajša projekta
- izkušnje ljudi!
- part-time ljudje, npr. študenti

Ocene stroškov: v fazi načrtovanja in tudi med izvedbo se izboljšujejo!
V fazi definiranje projekta: ocenimo red velikosti stroškov: ocena, ki je lahko 75% nižja ali 25% višja
Med načrtovanjem: groba ocena, ki je lahko 25% nižja ali 10% višja
Med izvajanjem, za naslednjo aktivnost: definitivna ocena, ki je lahko 10% nižja ali 5% višja
Kontrola stroškov, sproti običajno predrago, ob rednih časovnih intervalih, tedensko, mesečno.

Poglavje 7

Izdelava in analiza mrežnega diagrama

Tretja in zadnja točka v fazi Načrtovanja projekta!

Mrežni diagram je vizualna predstavitev zaporedja aktivnosti v projektu.

Najpreprostejša definicira projekta je zaporedje povezanih aktivnosti, ena po ena, dokler projekt ni končan.

To običajno zahteva preveč časa saj aktivnosti, ki se lahko izvajajo hkrati oziroma vzporedno, skrajšajo čas celotnega projekta.

Najprej je potrebno ugotoviti, kako so aktivnosti med seboj odvisne:

- pogled naprej: katere aktivnosti morajo biti končane, preden se lahko začne druga aktivnost?
- pogled nazaj: katere aktivnosti bi se lahko začele, če je skupina aktivnosti končana?

7.1 Struktura aktivnosti

Ganttov diagram (začetek 20. stoletja) pokaze predvsem, kako si aktivnosti v projektu sledijo

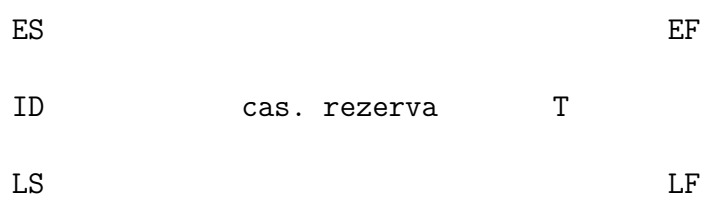
za zelo preproste, kratke projekte

slabost ganttovih diagramov: odvisnost aktivnosti ni dobro razvidna

Mrežni diagram (po 2. svet. vojni): jasna vizualizacija razmerij med aktivnostmi

uporaben za nacrtovanje, spremljanje in kotrolo izvajanja projekta

iz mreznega diagrama lahko generiramo ganttov diagram!



Slika 7.1: Aktivnost v aktivnostnem diagramu: ES - earliest start, EF - earliest finish, LS - latest start, LF - latest finish, T - čas trajanja

Zgodovina: AOA (Activity on arrows) diagrami (Polaris rakete), na racunalniku pa raje AON (Activity on nodes) diagrami.

Za vsako aktivnost rabimo spisek njenih predhodnih in naslednih aktivnosti.

Vrste odvisnosti med aktivnostmi (S – Start, F – Finish)

- FS: ko A konca, se zacne B
- FF: ko A konca, lahko konca B
- SS: ko A zacne, lahko zacne B
- SF: ko A zacne, lahko B konca

FS je zazeljena odvisnost, ker je najbolj enostavna!
druge odvisnosti vpeljujemo le, če je to potrebno za skrajsevanje projekta!
Vzroki za odvisnosti med aktivnostmi:

- tehnicne odvisnosti
  - ocena vodje projekta (ker pozna ljudi, ker bo lažje vodil)
  - primeri dobre prakse
  - logicno razmisljanje
  - izjemne (unikatni kos opreme), edini strokovnjak
- upravljalske odvisnosti

  zaradi konkurence se mudi
- medprojektne odvisnosti

  projekt je del vecjega projekta

- datumske odvisnosti

  ceprav mocno odsvetovane in nezazeljene, vcasih so:

  ne prej kot

  ne kasneje kot

  tocno na ta dan

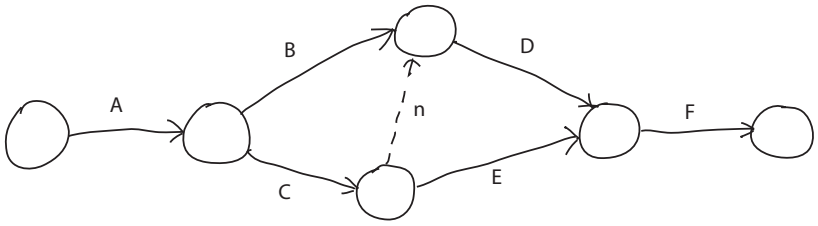
concurrent engineering: hitreje vendar bolj rizično!

Casovni zamik v odvisnosti med aktivnostmi se naredi takrat, ko nekaj čakamo in ne delamo nič.

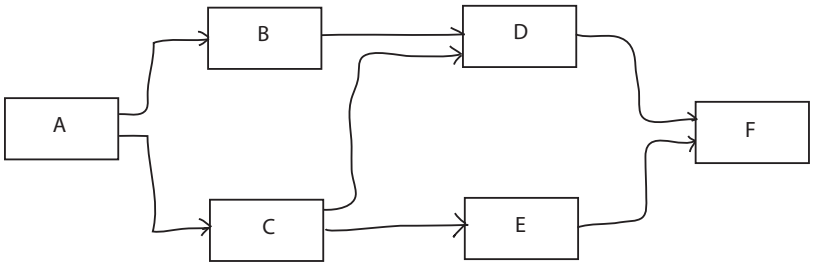
7.2 Izdelava začetnega mrežnega diagrama

Tabela 7.1: Primer preprostega projekta s šestimi aktivnostmi

aktivnost	neposredna predhodna akt.
A	-
B	A
C	A
D	B, C
E	C
F	D, E



Slika 7.2: Dogodkovni diagram: v vozliščih grafa so dogodki! Potrebna je navidezna aktivnost n, ki ima čas trajanja nič!



Slika 7.3: Aktivnostni diagram: v vozliščih grafa so aktivnosti!

uporabi le FS odvisnost, po mrezi naprej za izracun najzgodnejsih casov in po mrezi nazaj za izracun najkasnejsih casov

Za vsako aktivnost dobimo casovno okno v katerem se mora zaceti in dokoncati aktivnost!

Zaporedje aktivnosti, ki doloca celotni cas projekta - kriticna pot!

7.3 Analiza mrežnega diagrama

ce je cas trajanja projekta OK, potem OK

Če zelimo krajsi projekt:

cas projekta in potrebne zmogljivosti so medsebojno odvisne, vendar aktivnosti in resurse za njih ocenjujemo vsako posebej.

Krajsanje projekta izvedemo pod predpostavko, da je zmogljivosti dovolj.

Ce je projekt predolg, niti ni vazno ali je zmogljivosti premalo ali prevec oziroma neenakomerno obremenjene

7.4 Krajšanje projekta

analiziramo mrežni diagram

kjer je mozno, zaporedne aktivnosti izvajamo vzporedno, FS v SS

osredotocimo se na kriticne aktivnosti, to je aktivnosti na kritični poti,

s tem povecujemo tveganja, ker več aktivnosti lahko postane kritičnih

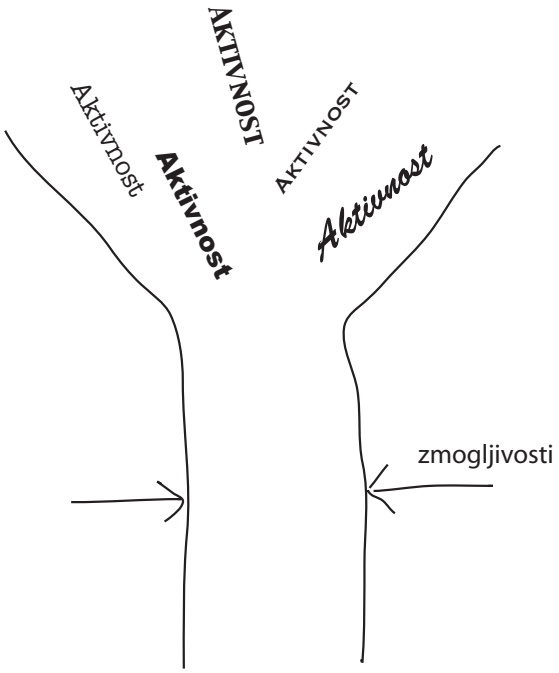
s spreminjanjem vrste odvisnosti se lahko spreminja kriticna pot

managerjeva rezerva, namesto casovnih rezerv v posameznih aktinostih bolje rezervna aktivnost na koncu (5 do 10% casa)

7.5 Kako v praksi izdelati mrezni diagram?

tezko veliko aktivnosti hkrati gledamo na racunalniskem zaslonu
vnesemo aktivnosti v program, natisnemo in izrezemo posamezne aktivnosti
nalepimo na post it listke in jih nalepimo na tablo
aktivnosti nato lahko razporejamo po tabli in hkrati diskutiramo
vnesemo rezultate v program

7.6 Optimizacija mrežnega nacrta glede na zmogljivosti



Slika 7.4: Mrežni diagram je potrebno uskladiti glede na razpoložljive zmogljivosti

izravnavanje zmogljivosti je potrebno
ce tlacimo notri prevec, pride do turbulenc, zamud, slabse kvalitete

ker je sirina lija fiksna, je mozno v dolocenem casu narediti le doloceno kolicino dela.

izravnavanje zmogljivosti je potrebno ker:

- zmogljivost ne more biti obremenjena vec kot 100% razpolozljivega casa
- stevilo zmogljivosti mora slediti tipicni razporeditvi zmogljivosti preko projekta (malo veliko malo)

ali pa divje fluktuirati od tedna do tedna

zmogljivosti niso sposobne delati 100% delavnega casa na neki aktivnosti, tipicno 50 do 65%

nadure so mozne, vendar ne ze v casu planiranja, raje izhod v sili kot oblika rezerve

7.7 Strategije za izravnavo zmogljivosti

- Uporaba obstojecih casovnih rezerv
- Podaljsanje projekta
- Posamezne spice v histogramu porezemo z nadurami
- delitev aktivnosti na krajse, vmes se ne dela
- podaljsevanje aktivnosti, z manjso izrabo delavnega casa, da bi ohranili kontinuiteto aktivnosti
- ce ni neke zmogljivosti, uporaba nadomestne, manj izkusene, podaljsanje obicajno potrebno

7.8 Vpliv izravnavanje na stroske

ce se projekt podaljsa:

ce se delo placa na osnovi opravljenega dela, se stroski ne povecajo

stroski se povecajo, ce jih placujemo sorazmerno s casom

7.9 Vodenje projektov na mikro ravni

Work package:

- opis, kako se naj aktivnost izvede

- najveckrat je na nivoju posamezne aktivnosti le spisek nalog, ki jih naredimo v poljubnem vrstnem redu
- lahko pa tudi naredimo mrezni diagram nalog, ce so naloge prepletene
- za projekt naredi pisek vseh work packagov (kdo je odgovoren, kdaj se zacne, kdaj konca)
- za vsak work package dokument z opisom nalog

Poglavje 8

Projektna skupina

VLOGE: naročnik projekta, projektna skupina, vodja projekta, svetovanje in preverjanje

Projektna skupina kot vrsta skupine: sekundarna, formalna, za reševanje problemov, prej majhna kot velika skupina,

Delez komuniciranja na posameznika vedno manj enakopravno, ko se skupina veča.

Primarna naloga vodje projekta je, da dokonča projekt pravočasno, v okviru budžeta in skladu s specifikacijo.

RAZLIKA! Ena od glavnih nalog vodje funkcionalnih enot pa je razvoj kadrov.

Z vidika učinkovitosti projektov bi si želeli delati le z najbolj izkušenimi ljudmi, vendar kje naj ljudje pridobivajo izkušnje?

Dobro je tudi uporabiti ljudi za naloge izven njihovih standardnih delovnih izkušenj.

8.1 Kaj motivira ljudi

Higienski faktorji (če jih ni, demotivacija, samo po sebi pa ne stimulirajo)

- politika podjetja
- administracija
- delovni pogoji
- tehnični nadzor
- medosebni odnosi

- varnost delovnega mesta
- plača

Motivatorji:

- dosežek
- priznanje
- napredovanje in rast
- odgovornost
- delo samo (ni monotono, poistovetenje z delom, pomembnost dela, avtonomnost, feedback)

8.2 Sestava projektne skupine

1. Vodja projekta
2. notranji člani projektne skupine
3. zunanji člani projektne skupine

8.2.1 Vodja projekta

KDAJ naj bo izbran?: idealno na samem začetku projekta, časih šele, ko je projekt potrjen

Prej ko so vodja in drugi člani projektne ekipe izbrani, bolj se poistovetijo s projektom!

Lastnosti vodje:

- izkušnje in znanje s področja vodenja projektov
- vodstvene lastnosti, strateške izkušnje
- tehnična znanja z področja projekta

Pri vodenju se prepletata dve skupini nalog:

- vodstvene naloge v zvezi z organiziranjem in vodenjem skupine, administriranjem in stiki z naročniki

- tehnične naloge v zvezi s samo naravo dela, izbira metod dela in pravilnih tehničnih rešitev.

Obe skupini nalog sta lahko združeni v eni osebi, sicer delitev na: tehnik/manager, režiser/producent

druge kompetence vodja: medosebni odnosi, reševanje konfliktov

diktator ali demokrat?

8.2.2 Člani projektne skupine

Ali lahko vodja projekta sploh izbira člane skupine?

Željene karekteristike:

- predanost
- odgovornost
- fleksibilnost
- izpolnjevanje ciljev
- zaupanje in podpora
- delo v skupini
- iznajdljivost, sposobnost komuniciranja

8.2.3 Zunanji člani

npr. študentsko delo

Slabosti: potreben večji nadzor, manj predanosti

Prednosti: jih najamemo le takrat, ko so potrebni, npr. za špice, slabše plačani?

8.3 O skupinskem delu

Ko se skupina ustanovi, sledi obdobje konfliktov, nato se poveča kohezivnost. S konflikti člani skupine preizkušajo omejitve formalnih in neformalnih pravil obnašanja.

Homogene/heterogene skupine (starost, znanje, spol, status). Katere so bolj učinkovite?

Koordinacijski mehanizmi:

1. preprosta struktura — direktni nadzor (delavci, delovodja)
2. delovna birokracija — standardizacija delovnega procesa (tekoči trak)
3. delitvena oblika — standardizacija delovnih rezultatov (obrtnik)
4. profesionalna birokracija — standardna izobrazba, licenca (zdravniki, pravniki)
5. pripadnost — vzajemno prilagajanje

Način vodenja: glede na medsebojni odnos in odnos do dela:

Tabela 8.1: Štirje osnovni načini vodenja ljudi

		odnos do dela: nizek	odnos do dela: visok
medsebojni odnos	nizek	ločen način	požrtvovalen način
	visok	povezan način	integracijski način

Poglavje 9

Spremljanje in nadzorovanje izvajanja projekta

ZAKAJ:

1. sledenje napredka
2. ugotavljanje odstopanj od nacrta
3. izvajanje korekcij

vec kontrole, manj tveganj
manj kontrole, vec tveganj
prava mera, pozitivna korelacija s kvaliteto

Slika 9.1: Cena kontrole in tveganja se seštevata!

Poročilo naj ima naslednje lastnosti:

- pravočasno, popolno in natančno poroča o stanju projekta
- ne zahteva preveč časa za pripravo, saj bi to bilo kontra produktivno
- je sprejemljivo za projektno skupino in vodstvo
- pravočasno opozarja na probleme, tako da je možno reagirati
- je razumljivo vsem, ki se morajo seznaniti z njim

Vrste porocil:

- za zadnje tekoče obdobje
- kumulativno od začetka projekta
- poročila o odstopanju od načrta
- z barvami semaforja opozarjamo če je OK (zelena), manjši problemi (rumena), resni problemi (rdeča).

  ZAKAJ: odstopanja opazimo čimprej
- odstopanja:

  numericno poročanje (načrtovana številka, dejanska, razlika),

  graficne tehnike: lažje razberemo trende

Katere podatke poročati:

- definirati casovni trenutek, ko je potrebno podati porocilo
- dejansko opravljeno delo v obdobju poročila
- dejanski datumi zacetka in konca aktivnosti
- koliko dni se že dela na aktivnosti in nova ocena koliko dni do konca
- % dejanskega dela v delavnem času
- % dokoncanja (čas, dela, stroškov)

Običajna frekvenca poročanja je tedenska.
Pozitivna in negativna odstopanja.

9.1 Grafične tehnike

- ganttovi diagrami
- Milestone trend chart: zamude in prehitevanje mejnikov v projektu
- stroski (S krivulja): primerjava načrtovane porabe sredstev z dejansko porabo, manjša poraba ponavadi pomeni, da aktivnosti zamujajo!
- z uporabo WBS diagrama: označimo aktivnosti, ki so delno ali povsem končane

- podrobnosti porocil odvisne od komu je namenjeno:
  - vodje posameznih aktivnosti: najbolj podrobno
  - vodja projekta: podatki o posameznih aktivnostih in kako te vplivajo na celoten projekt
  - vodstvo: kratko, jedrnato, ganttovi diagrami

sestanki, kjer se poroča:

Kdo se jih naj udeležuje? ne preveč ljudi, podrobnosti naj se obravnavajo na satelitskih sestankih

Kdaj? ob koncu delavnega tedna

Kaj je njihov namen? da se informira celotna projektna skupina, zato se mora objaviti zapisnik

9.2 Upravljanje s spremembami

Vsaka zahteva za spremembo mora biti dokumentirana!

Odražati se mora v COS (Conditions of Satisfaction)!

Zahteva se mora analizirati in napraviti poročilo o vplivu spremembe (pozitivne, negativne).

Možnih je 6 odgovorov na zahtevo:

1. možno jo je upoštevati v okviru prvotnih zmogljivosti in časa projekta
2. možno jo je upoštevati, vendar se bo čas projekta podaljšal
3. možno jo je upoštevati v danem časovnem okviru, vendar bo zahtevala dodatne resurse
4. možno jo je upoštevati vendar se čas podaljša in resursi povečajo
5. možno jo je upoštevati, vendar s postopno realizacijo ciljev
6. ni je možno upoštevati brez velikih sprememb projekta

Poglavje 10

Zaključevanje projekta

1. Naročnik sprejme rezultate: formalno sprejetje,

   včasih de facto, če se je nekaj odvilo oziroma končalo (npr. organizacija konference, konferenca se je odvila, ne glede na to, ali so bile izpolnjene vse zahteve v specifikaciji)

2. Vsi rezultati projekta so inštalirani

3. Vsa dokumentacija je urejena: koristno za večih vidikov (za morebitno kasnejše spremembe, za kalibriranje naših postopkov, izobraževanje)

4. Naročnik podpiše končno poročilo

5. Analiza končanega projekta: kaj smo se naučili, da bi pri prihodnjem projektu bili boljši?

6. Praznovanje uspeha!

Poglavje 11

Adaptivno vodenje projektov

V sodobnem poslovnem svetu je veliko hitrih sprememb.

V TPM načinu se pogosto izdela natančne načrte za aktivnosti, ki jih nikoli ne izvajamo.

Veliko časa in resursov se na ta način zapravi.

Pri adaptivnem vodenju projektov se stalno prilagajaš spremembam in zahtevam naročnika.

Načrtovanje se izvaja po principu **just in time**.

Na začetku se izvede načrtovanje le na nivoju glavnih funkcij oziroma komponent (2 do 3 nivoji v WBS), kako se bodo te funkcije implementirale pa se načrt naredi tih pred njihovo implementacijo.

Rezultate projekta se gradi iterativno skozi več ciklov.

Dolčina posameznega cikla je tipično 2 do 6 tednov.

Število ciklov je odvisno od tega koliko jih gre v zamišljeno trajanje projekta.

Ob koncu vsega cikla analiziraj narejeno delo in naredi načrt za naslednji cikel.

Adaptivno vodenje projektov ima 5 faz:

11.1 Določanje obsega projekta

Kot vsak projekt se začne z reševanjem problema ali identificirano poslovno priložnostjo.COS dokument se razvije na osnovi razgovora med izvajalcem in naročnikom. Odločimo se na kakšen način vodimo projekt, TPM ali AVP.

Najprej napišemo POS dokument, ki rezimira COS in vsebuje naslednje točke:

1. Kaj je problem ali priložnost (razlogi za projekt)?
2. Identifikacija cilja (kaj hočemo narediti)?

3. Možni načini doseganja tega cilja?

4. Kaj bi doseženi cilj pomenil v poslovnem smislu, kriteriji za merjenje uspešnosti projekta?

5. tveganja, ovire, predpostavke

Druga stvar, ki jo naredimo v tej prvi fazi je seznam funkcij, ki jih želimo realizirati.

Tretja je WBS diagram (2 do 3 nivoji), ki vsako želeno funkcijo razdeli vsaj na podfunkcije. Ne želimo pa že načrtovanti, kako bodo te funkcije realizirane.

Četrti element, ki ga izdelamo v tej fazi je prioritetni vrstni red spremenljivk v trikotniku soodvisnosti (čas, cena, zmogljivosti, v sredini pa obseg in kvaliteta).

11.2 Načrtovanje posameznega cikla razvoja

Načrtovanje je tako preprosto, da ni potrebna uporaba programske opreme.

Običajno več majhnih skupin vzporedno implementira vsaka svoj kos funkcionalnosti. Zato vsaka skupina načrtuje in nato implementira oziroma izvaja svoj načrt.

Dolžina cikla je fiksna in se ne podaljšuje!

11.3 Izvajanje posameznega cikla

Vse načrtovanje funkcionalnosti, ki niso implementirane v tem ciklu, jih lahko vključimo v naslednji cikel.

11.4 Kontrola posameznega cikla

Preverjanje rezultatov doseženih v posameznem ciklu se naredi skupaj z naročnikom!

Izvajalci in naročniki skupaj preverijo rezultate zadnjega cikla. Primerja se jih z globalnim cilje in po potrebi se naredi spremembe globalnega načrta in načrt naslednjega cikla.

Zaporedje: načrtovanje cikla, izvajanje cikla in kontrola cikla se ponovi dokler nismo dosegli časovnih in stroškovnih omejitev za projekt.

Vso novo znanje in izkušnje, ki smo jih dobili v enem ciklu, lahko uporabimo v naslednjih ciklih.

Odkrili bomo morda alternativne načine za implementacijo in doseganje ciljev.

V naslednji cikle načrtujemo le implementacijo tistih funkcionalnosti, ki bodo zanesljivo v končni rešitvi.

Vključevanje naročnika je izjemno pomembno.

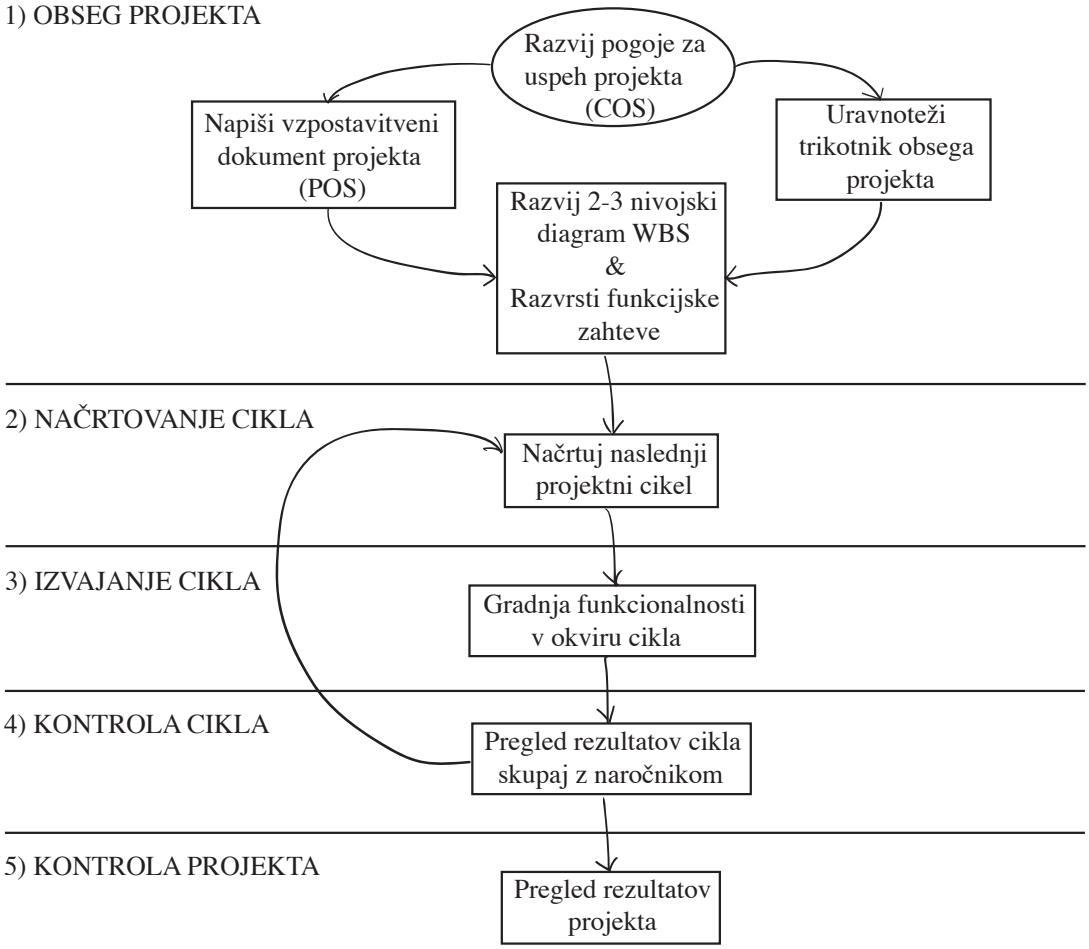
11.5 Končna kontrola projekta

Merilo za uspeh projekta so kriteriji, ki smo jih navedli v POS dokumentu.

Porabili smo le toliko časa in denarja, kot smo prvotno načrtovali.

Pri večjih in daljših projektih težje priznamo poraz in ga hočemo rešiti z več resursi in časa.

Projekt lahko končamo po kateremkoli ciklu.



Slika 11.1: Adaptivno vodenje projektov ima pet korakov. Celoten projekt naj bi trajal maksimalno 6 mesecev, posamezen cikel pa 2 do 6 tednov.

Poglavje 12

Lastnosti adaptivnega vodenja projektov

1. osredotoceno na narocnika, potrebe naročnika so najpomembnejše,
2. narocnik lahko direktno vpliva na potek projekta
3. inkrementalni rezultati zgodaj in pogosto nam omogoča, da lažje najdemo najpomembneše funkcionalnosti.

   prvi cikel je lahko le demonstracija koncepta (proof of concept).
4. stalno sprasevanje in analiza omogoča učenje in raziskovanje ter prilagajanje
5. spremembe kot pot k boljsi resitvi, po vsakem ciklu vemo več o problematiki in lahko predlagamo boljše rešitve
6. izognemo se lahko vsem delu, ki ne prispeva h končnemu cilju (nacrtovanju za aktivnosti, za katere ni jasno, da se bodo izvajale)

Poglavje 13

Določanje obsega projekta

Ta faza je sestavljena iz dveh delov: definiranje projekta in načrtovanje projekta.

13.1 Definiranje projekta

Pri definiranju je pomembno, da sodelujejo tako izvajalci in naročniki projekta. Načrtovanje se sicer začne kot pri TPM-ju, vendar ni tako podrobno.

Bistvena razlika med TPM in AVP je pri obravnavanju sprememb. Pri TPM spremembe tipično pomenijo več časa, več zmogljivosti in več denarja.

Pri AVP pa je možno spremembe, ki so rezultat učenja izvajalcev in naročnikov vključiti v projekt in projekt še vedno končati v roku in v okviru načrtovanih stroškov.

1. dokument COS (Conditions of satisfaction):

   Izdelava COS dokumenta naj bo narejena v majhni skupini, ki lahko sprejema odločitve.

   Za COS sta potrebni dve vrsti pogovora:

   najprej govori naročnik, opisuje zahteve tako kot jih on razume. Izvajalec lahko postavlja vprašanja in ponovi zahteve tako kot jih on razume. Naročnik naj bi na koncu povedal izvajalcu: Dobro me razumete, kaj si želim od vas!

   Nato govori izvajalec: pove in opiše kaj lahko naredi, da bi izpolnil naročnikove zahteve. Naročnik lahko sprašuje in ponovi odgovore izvajalca, tako kot jih on razume. Na koncu naj bi izvajalec povedal: Jasno mi je, kaj lahko naredim!

   Ta komunikacija je osnova za COS. V tem trenutku še ne vemo kakšna bo rešitev

2. dokument POS (Project overview statement): 5 elementov (problem/priložnost, cilj, močne poti do cilja, kriteriji za uspeh, tveganja in ovire) dodatki (analize tveganj, ekonomske študije, tehni v cne študije).

   cilj naj bo raje majhen kot prevelik, bolje je izdelati več verzij projekta, saj so projekti na ta način bolj obvladljivi.

   možne poti bolj natančno opišejo, kako je možno cilje doseči

   kriteriji uspeha naj bodo merljivi!

3. Budget in čas za projekt naj bo fiksen!

   Čas naj ne bo daljši od 6 mesecev!

   Motiv AVP je maksimizirati poslovne kriterije v okviru fiksnega budžeta in časa!

13.2 Planiranje projekta

1. Naredi WBS do sredine (2-3 nivoji v hierarhiji)

   definiranje funkcionalnosti, ki bo narejena v tej verziji

2. Razvrsti potrebe po funkcionalnosti

   kriteriji za razvrstitev: tveganje, kompleksnost, trajanje, poslovna vrednost, medsebojne odvisnoti

   musts, shoulds, nice to

3. Razvrsti po pomembnosti trikotnik obsega (čas, stroški, zmogljivosti, kvaliteta+obseg)

   Trikotnik obsega je pri AVP model za odločanje: najpomembnejši parameter bomo spreminjali nazadnje!

   Število ciklov v verziji

   najprej pogosti, da se hitro vidijo rezultati,

   glede na razvrstitev funkcionalnosti, jih razvrsti v cikle

   jasno napiši, kaj lahko pričakujemo od vsakega cikla, važno za naročnika

Poglavje 14

Planiranje posameznega cikla

tu se veckrat ponovi zanka, ki se vraca iz preverjanja cikla

1. Iz WBS izloci aktivnosti/funkcionalnosti, ki se bodo zgradile v tem ciklu
2. dodatno razdeli WBS za te aktivnosti do nivoja aktivnosti oziroma nalog (task)
3. oceni potreben čas in resurse za vsako nalogo
4. ugotovi odvisnosti med nalogami
5. razdeli naloge na skupine nalog in za vsako skupino doloci tim ali posameznika
6. vsak tim naredi mikro mrezni nacrt znotraj casovnega in stroskovnega okvira

ne uporablja se klasicno mrezno nacrtovanje in programska oprema
ročno načrtovanje cikla s pomočjo post-it listkov

Poglavje 15

Gradnja posameznega cikla

gradnja oziroma izvajanje cikla poteka le toliko casa, kot je bilo nacrtovano

funkcionalnost, ki ni bila zgrajena, se zgradi v naslednjem ciklu

če pride to problemov skušamo dokončati cikel, sicer pa lahko cikel predčasno prekinemo in gremo takoj v kontrolo (preverjanje) cikla in načrtovanje naslednjega cikla

15.1 Nadzorovanje in prilagajanje cikla

Za nadzorovanje in prilagajanje izvajanja cikla projektna skupina uporablja 3 orodja, ki morajo biti vidna vsem članom projektne skupine:

1. seznam idej

   Pri vsakem delu hkrati poteka proces odkrivanja in učenja. Projektna skupina dobiva nove ideje v zvezi z cilji projekta ali z načinom, kako te cilje uresničiti.

   Vse te ideje je potrebno zabeležiti in jih obravnavati v naslednji fazi Kontrole cikla.

2. seznam problemov

   Pri vsakem delu pride to težav in problemov. Vse te težave je potrebno zapisati, najbolje kar z roko na tablo, saj je tako najlažje osveževati seznam.

3. matrika posledic

   Pri seznamu idej (spremembe ali dodatne funkcionalnosti) je potrebno za vsako točko oceniti, kakšno posledico bi ta imela za projekt.

Za vsak problem v seznamu problemov pa kakšne potencialne posledice problem ima, če ga pravočasno ne razrešimo.

Projektna skupina naj bi imela kratek sestanek vsako jutro (15 minut). Poroča se o statusu, problemih in rešitvah problemov. Ni diskusije, po potrebi se osveži seznam idej in problemov. Če je skupina večja, potem se sestajajo le vodje skupine nalog.

Poglavje 16

Preverjanje rezultatov cikla z naročnikom

zelo pomembna faza, mozno integrirati vse spremembe,

Vhodne informacije za preverjanje rezultatov cikla sta:

1. načrtovana in dejansko implementirana funkcionalnost v pravkar končanem ciklu

   Ker je dolžina posameznega cikla fiksna, je možno, da nismo mogli implementirati vse funkcionalnosti.

2. seznam idej

   Seznam idej je kumulativen repozitorij vseh idej in predlaganih sprememb od samega začetka projekta. Nekatere je bilo možno upoštevati že v prejšnjih ciklih in zato niso več na seznamu. V seznamu so vse tiste, ki jih do sedaj še ni bilo možno upoštevati.

Vprašanja, ki jih moramo obravnavati pri preverjanju rezultatov skupaj z naroňikom:

1. Kaj je bilo načrtovanjo v pravkar končanem ciklu?

   To je enostavno seznam funkcionalnosti, ki smo ga načrtovali za obravnavani cikel.

2. Kaj je bilo narejeno?

   Odkljukamo to, kar smo naredili z opombami, če je prišlo do manjših odstopanj pri posameznih funkcionalnostih.

   Funkcionalnosti, ki jih nismo implementirali moramo sedaj upoštevati pri načrtovanju naslednjega cikla.

3. Ali je načrtovani obseg projekta še pravilen?

   Če je odgovor da, potem smo na dobri poti. Sicer pa se lahko odločimo tudi, da projekt prekinemo. Tak prekinjen projekt še vedno pomeni manjšo izgubo, kot če bi delali na tradicionalni projektni način, ko običajno projekt pripeljemo do konca in porabimo vsa načrtovana sredstva, rezultatov pa ni.

4. Ali je projektna skupina delovalo tako kot je bilo pričakovano?

   Pomembna je predvsem identifikacija članov projektne skupine s projektom. Vladati mora taka atmosfera, da se člani projektne skupine upajo odkrito spregovoriti o problemih.

5. Kaj smo se naučili?

   To je najbolj pomembna razlika s tradicionalnim vodenje projektov, kjer enostavno ni trenutka, kjer bi se to lahko vprašali.

Prilagajanje seznama funkcionalnosti za naslednji cikel:

Vhod

1. funkcionalnosti implementirane v prejšnjem ciklu,
2. funkcionalnosti načrtovanje za prejšnji cikel, vendar niso bile implementirane
3. funkcionalnosti načrtovanje za naslednji cikel
4. funkcionalnosti načrtovane za vse naslednje cikle
5. seznam idej
6. seznam problemov

Izhod

1. osvežen seznam funkcionalnosti:
   Ali COS še velja? Če ne, potem je potrebno osvežiti seznam funkcionalnosti.
2. nova razvrstitev vseh funkcionalnosti
3. dolžina naslednjega cikla

Poglavje 17

Zaključevanje projekta

Podobno zaključevanju tradicionalnih projektov.

Ali so bili doseženi poslovni cilji (kontrola s pomočjo COS)?

Kaj smo se naučili za naslednjo verzijo projekta?

Poglavje 18

Variacije na Adaptivno vodenje projektov

Adaptivno vodenje projektov je prilagodljivo, ker:

- se spreminja število ciklov
- spreminja se dolžina posameznega cikla
- na koncu vsakega cikla lahko spremenimo razvrstitev funkcionalnosti
- vključujemo lahko spremembe (dodajamo nove, spreminjamo ali ukinjamo obstoječe ob vsakem zaključku cikla

Adaptivno vodenje projektov predstavlja nek kuntinuum, kjer so na eni strani tradicionalni projekti (en cikel!), na drugi pa ekstremni.

18.1 Ekstremni projekti

Ekstremni projekti so:

- Hitri

  to so projekti, ki so inovativni, kritični za prihodnost podjetja, v temelju spremenjajoči, rezultati naj bi bili čimprej

- Vnašajo veliko sprememb

  med projektom poteka učenje in odkrivanje, spremembe med projektom so lahko še bolj radikalne kot pri adaptivnih projektih

- visoka negotovost

  ker so ekstremni projekti inovativni in raziskovalno usmerjeni, nihče ne ve, kam in kako se bodo razvili, tudi čas projekta ni vnaprej znan, veliko bo poskusov in napak, tudi veliko neuspelih projektov

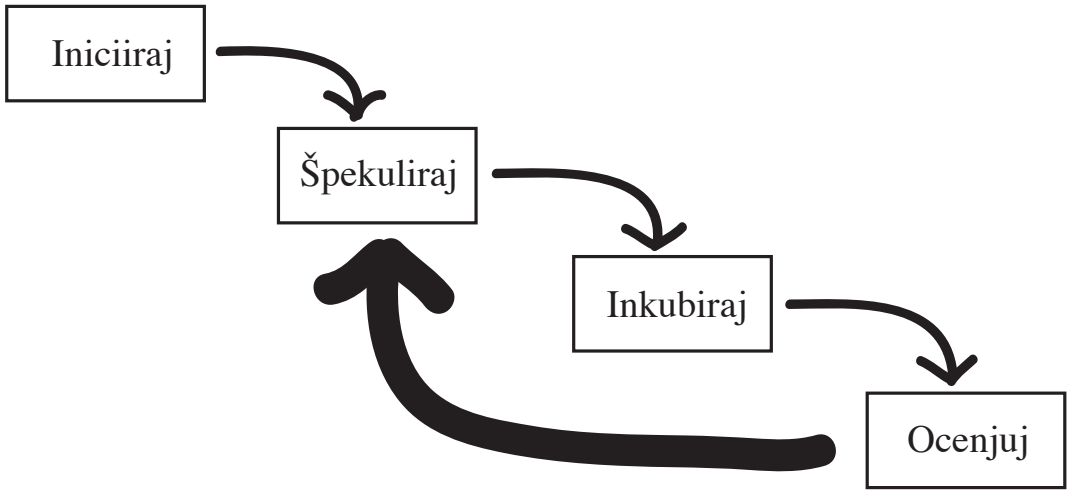
Po svoji naravi so ekstremni projekti nestrukturirani. Učenje in raziskovanje poganjata ekstremni projekt naprej.

Esktremni projekt pa tako kot adaptivni način vodenja uporablja iteracije.

Vloga stranke je še bolj poudarjena, potrebna je tudi med samimi cikli in ne le ob koncu ciklov kot pri adaptivnem vodenju.

Za razliko od adaptivnega vodenja omejitve, ki jih postavlja trikotnik obsega, pri ekstremnem vodenju projekta niso smiselne.

Delamo dokler ne dosežemo cilja ali pa zmanjka časa ali denarja. Kljub temu veljajo običajne povezave med parametri v trikotniku obsega. Od projekta do projekta je odvinso, katerega od parametrov je smiselno najprej prekoročiti. Pri vsaki fazi projekta se lahko odločimo bodisi končati projekt bodisi zagotoviti več denarja in časa, če so dotedanji rezultati dovolj obetajoči.



Slika 18.1: Ekstremno vodenje projektov

18.1.1 Iniciiraj

Tu gre za to, da sponzorja projekta navdušimo za neko idejo, da tuhtamo o različnih možnostih, da najdemo poslovni smisel projekta.

Cilj projekta je pri ekstremnem vodenju bolj vizija neke bodočnosti (kaj si želim bom vedel, ko bom to videl). Je bolj kot neko avanturistično potovanje, ko destinacija ni točno določena. Cilj se oblikuje med samim potovanjem oziroma delom na projektu.

POS dokument za ekstremni projekt:

1. problem ali priložnost kot sicer pri projektih

2. definicija cilja: ni konkretnega cilja, oziroma cilja ne moremo časovno opredeliti.

3. možni načini doseganja cilja: ni možno slediti formata SMART. Naštejemo nekaj objektivnih dejavnikov, ki na široko določajo smer raziskovalnja. Ker pa cilj ni jasen, se ti dejavniki lahko v toku projekta spremenijo.

4. Kriteriji uspeha: postavimo le širši okvir okoli široko zastavljenega cilja

5. Tveganja, ovire, predpostavke: podobno kot pri TMP in adaptivnem vodenju projektov.

18.1.2 Špekuliraj

Ta faza predstavlja začetek novega cikla in se vedno začne z nekim "brainstorming" sestankom. Sodelujejo naj projektna skupina, naročnik in končni uporabnik. Določiti je potrebno dolžino naslednjega cikla in kaj bomo naredili v naslednjem ciklu.

POS je za prvi cikel dobro izhodišče, ko spoznavamo naše problemsko področje pa ga je potrebno posodavljati.

Uporabiti je potrebno scenarije, zgodbe in uporabniške primere za to kar želimo razviti.

Zahteve je potrebno razvrstiti.

Glede na to, kako dobro razumemo cilje, je v prvem ciklu v ekstremnem projektu smiselno raziskati čimveč različnih opcij in alternativ. Podobno je smiselno v prvem ciklu zgraditi prototip, da bi lahko nato bolje razvrstili funkcionalnosti v končnem produktu.

Kot smo zahteve razvrstili, se lahko odločimo kakšen delež jih bomo realizirali v prvem ciklu.

Na splošno želimo prve cikle čimkrajše, saj so spremembe v prvem delu projekta največje in da naročnika bolj motiviramo. Ker so začetni cilji bolj raziskovalni, se lahko šele takrat odločamo, ali sploh nadaljevati s projektom.

Včasih se moramo odločati ali gremo pri zahtevah najprej v globino ali v širino.

Vsaka nadaljna faza naj bi bila bolj določena, saj naj bi z vsakim ciklom več vedeli o projektu in njegovih ciljih.

18.1.3 Inkubiraj

Pri adaptivnem vodenju se v fazi izdelave oziroma realizacije funkcionalnosti v posameznem ciklu držimo načrta za ta cikel.

Pri ekstremnem vodenju pa dopuščamo spremembe tudi v samem ciklu.

Določiti je potrebno kateri resursi (ljudje) bodo delali na katerih funkcionalnostih. Časovni načrt se izdela podobno kot pri adaptivnem vodenju predvsem s pomočjo post-it listkov in risanja po tabli.

Sodelovanje je pri ekstremnih projektih izrednega pomena, saj je določanje cilja in način njegove realizacije potrebno šele določiti.

18.1.4 Ocenjuj

Ta faza je podobna kot pri adaptivnem vodenju. Analizirati je potrebno vse vidike zadnjega cikla projekta:

- kaj smo se naučili?

  zgodaj v projektu so običajne velike spremembe, kasneje pa naj bi stvari konvergirale

- kaj lahko naredimo da bi povečali možnost doseganja cilja?

  revizija cilja: katere alternative lahko eliminiramo, ali je potrebna spremeba smeri raziskovanja, ali stvari konvergirajo?

- katere nove ideje so se pojavile in ali naj jih zasledujemo?

- kaj naj naredimo v novem ciklu?

  ponovno je potrebno razvrstiti zahteve oziroma funkconalnosti, po potrebi spremenimo POS

  ali naj sploh nadaljujemo?

Tabela s primerjavo treh načinov vodenja projektov.

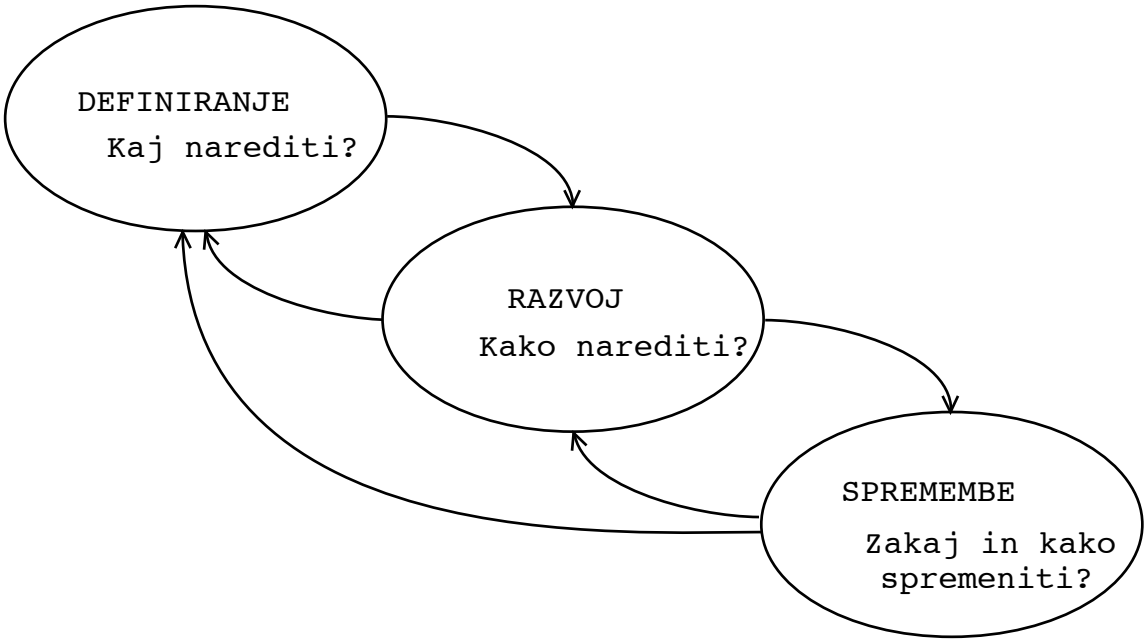
Poglavje 19

Primerjava tradicionalnega, adaptivnega in ekstremnega vodenja projektov

Tradicionalno	Adaptivno	Ekstremno
en cikel	določeno št. ciklov	nedoločeno št. ciklov
fiksen budget in čas	fiksen budget in čas	variabilen budget in čas
fiksen obseg	spremenljiv obseg	neznan obseg
popoln WBS	2-3 nivojski WBS in Just-in-time WBS	ni WBS
popoln načrt	Just-in-time načrt	Just-in-time načrt
netoleranten do sprememb	spremembe zaželjene	spremembe nujne

Poglavje 20

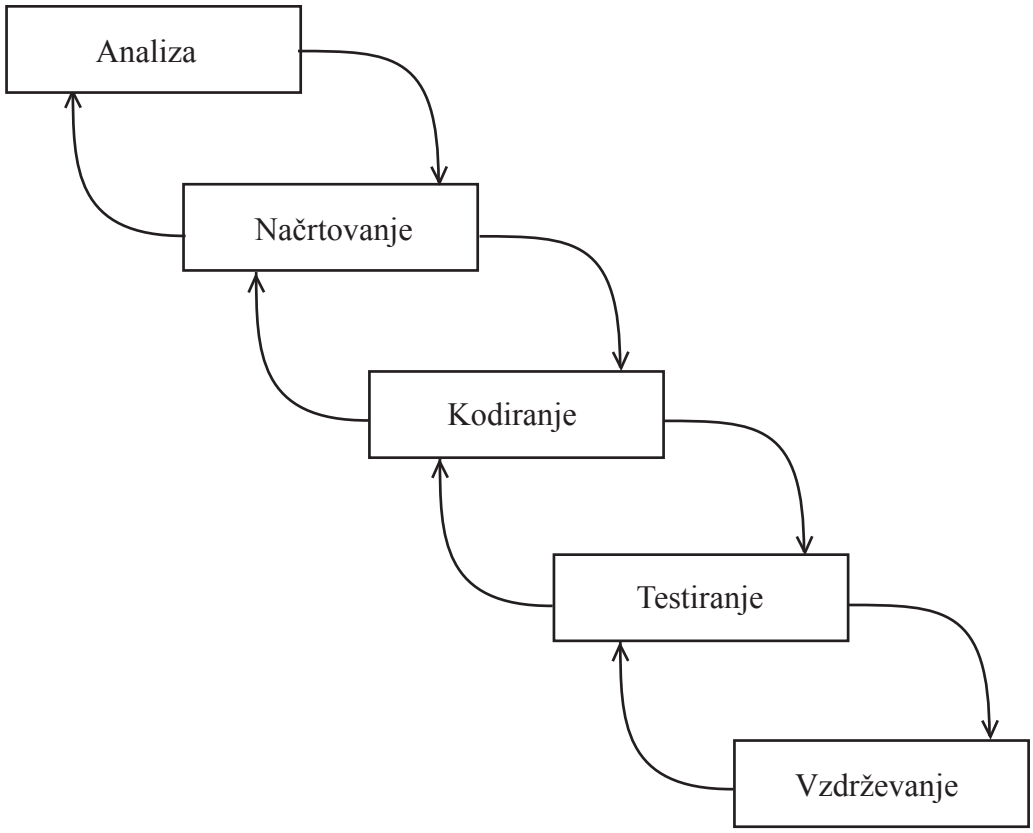
Modeli razvoja programske opreme



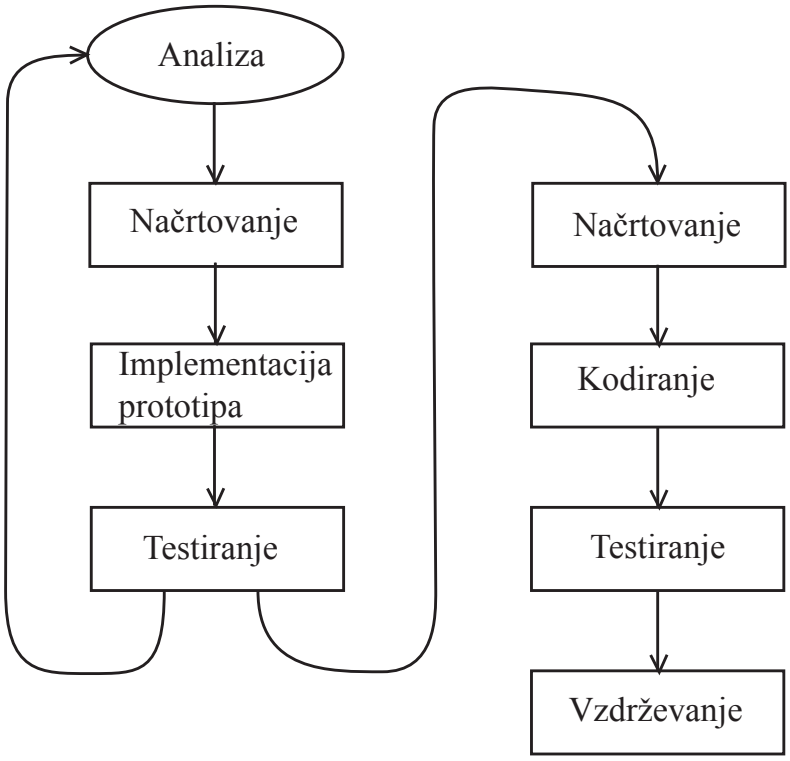
Slika 20.1: Generični življenski cikel programske opreme ima tri faze

20.1 Osrednje aktivnosti pri razvoju programske opreme

- **Analiza zahtev programske opreme.** Cilj analize je popolno razumevanje nalog in lastnosti programske opreme. Vse zahteve je potrebno skrbno dokumentirati in se o njih pogovoriti z naročnikom.
- **Načrtovanje** programske opreme je večstopenjski proces, ki zahteve s pomočjo uporabe podatkovnih in kontrolnih struktur ter programske arhitekture prevede v načrt programske opreme.
- **Kodiranje** je ob popolnem načrtu zgolj mehanske narave, saj naj bi že načrtovanje definiralo vse potrebne podrobnosti.
- **Testiranje** ni le izolirana aktivnost, ki preverja rezultate kodiranja, temveč se mora izvajati ves čas razvoja programske opreme.
- **Vzdrževanje** programske opreme je potrebno zaradi odpravljanja napak po končanem razvoju in zaradi spremenjenih zunanjih okoliščin, v katerih sistem deluje.



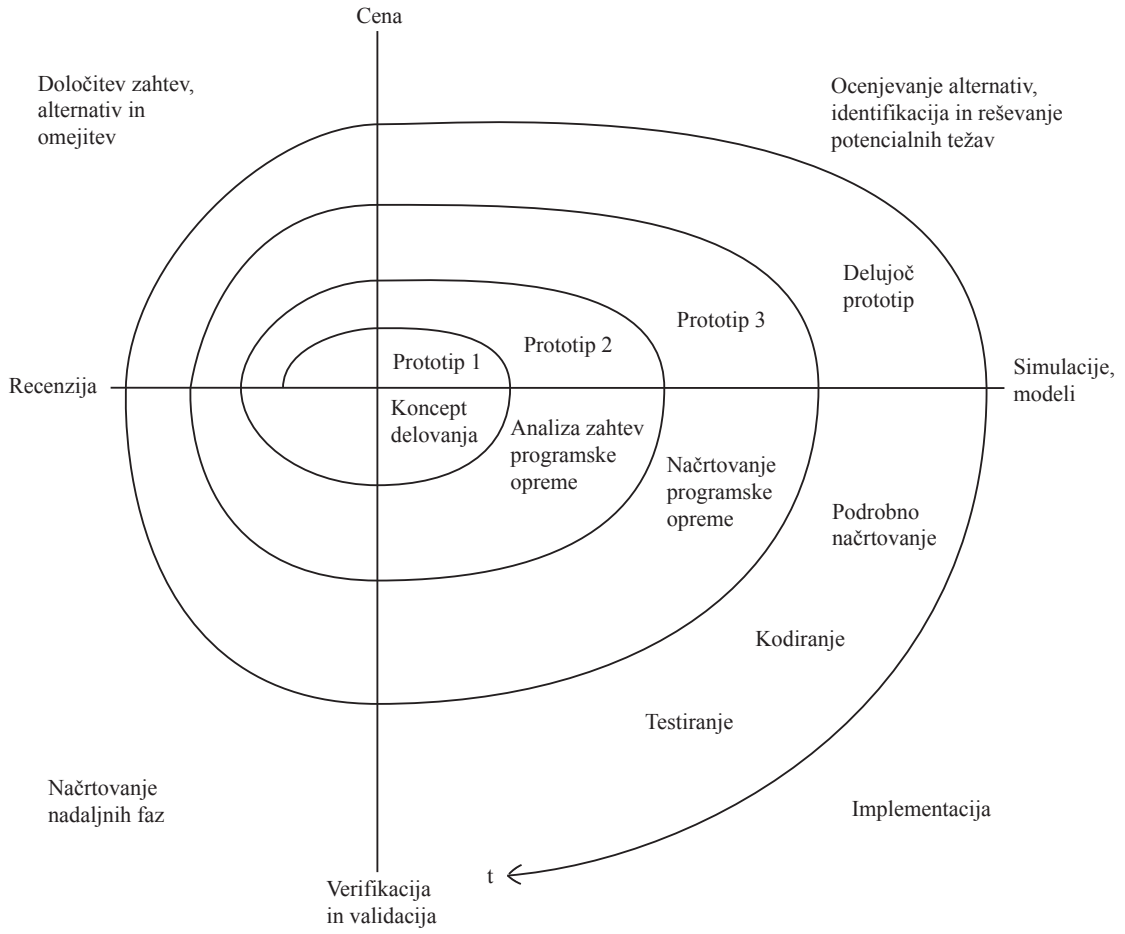
Slika 20.2: **Klasični ali kaskadni razvojni cikel** programske opreme. Primeren je le za take projekte, kjer je možno cilje oz. funkcionalnost programske opreme zelo natančno določiti na začetku projekta. Take projekte je možno voditi na tradicionalni način!



Slika 20.3: **Protopni razvojni cikel** programske opreme. Izdelava prototipa in njegovo ocenjevanje je namenjeno določanju zahtev in ciljev. Prototip običajno ni povsem funkcionalen, ampak osredotočen na tiste vidike, ki niso jasni. Prototip se sicer tudi lahko postopno razvije v končni produkt, bolje pa je, da služi le kot mehanizem za določanje zahtev. Primerno je adaptivno vodenje projektov.

Postopni razvoj programske opreme

- funkcionalnost produkta se postopoma širi
- med koraki postopnega širjenja funkcionalnosti se produkt že redno uporablja
- programska oprema postopoma raste in se vse bolj prilagaja uporabnikovim zahtevam
- uporabnika prisili, da funkcije razvrsti po pomembnosti



Slika 20.4: **Spiralni razvojni cikel** je meta model, ki vsebuje oziroma povezuje vse prej naštete modele razvoja programske opreme.

Poglavje 21

Značilnosti projektov za razvoj programske opreme

- projekti razvoja programske opreme se razlikujejo od drugih tehniških projektov
- razlikujejo se tudi med seboj glede na to, kako so organizirani, vodeni in kakšne cilje imajo:
  - stroški so vnaprej določeni in kvaliteto končnega produkta je potrebno maksimizirati glede na to omejitev,
  - kvaliteta programske opreme je predpisana in projekt mora čimbolj učinkovito zgraditi tak sistem,
  - nekateri morajo biti končani v minimalnem možnem času.
- delitev na ciljne, kontrolne in zunanje parametre ni stalna, saj je lahko nek parameter enkrat zunanji, drugič kontrolni ali ciljni:
  - če so cilji projekta (čas razvoja, kvaliteta) točno določeni, moramo zaposliti ustrezne ljudi in nabaviti potrebna orodja
  - če je delovna skupina fiksna, lahko kontroliramo projekt z daljšanjem časa razvoja ali manjšanjem kvalitete programske opreme
- za uspešno vodenje projekta mora biti vnaprej znano, katere parametre je možno spreminjati in z njimi kontrolirati potek projekta.

21.1 Vrste projektov

Stabilnost produkta je odvisna od tega, kako natančno so definirane zahteve po funkcionalnosti in kvaliteti.

Stabilnost razvojnega procesa je odvisna od stopnje kontrole, ki jo imamo nad razvojem, in od natančnosti, s katero lahko merimo razvoj.

Stabilnost zmogljivosti določa predvsem razpoložljivost kvalificiranih razvijalcev.

	Stabilnost		
Vrsta problema	produkta	razvoj. procesa	zmogljivosti
Problem realizacije	visoka	visoka	visoka
Problem razvrstitve	visoka	visoka	nizka
Problem načrtovanja	visoka	nizka	nizka
Problem iskanja	nizka	nizka	nizka

1. **Problem realizacije.** Če so zahteve jasne in stabilne, če vemo, kako naj teče razvojni proces in če imamo za povrh še dovolj zmogljivosti, lahko razvojni proces natančno kontroliramo. Osredotočimo se le na problem učinkovite realizacije (klasični razvojni cikel, direktni nadzor, ločitveni način vodenja, tradicionalni način vodenja projektov).

2. **Problem razvrstitve.** Nimamo dovolj zmogljivosti. Osnovni problem je, kako razvijalce razvrstiti na delovne aktivnosti. S standardizacijo razvojnega procesa je možno povečati izmenljivost ljudi med nalogami, kar je v taki situaciji izredno pomembno. Izberemo klasični razvojni cikel in tradicionalni način vodenje projektov.

3. **Problem načrtovanja.** Če so zahteve jasne in stabilne, ne vemo pa, kako jih naj uresničimo in zato tudi ne kakšne zmogljivosti potrebujemo. Zato moramo narediti ustrezen načrt: kateri so glavni mejniki v projektu, koliko ljudi potrebujemo in kako bomo vse skupaj kontrolirali. Kot razvojni model je nabolj primeren postopni razvoj in adaptivno projektno vodenje.

4. **Problem iskanja.** Če so zahteve nejasne, ni mogoče definirati niti razvojnega postopka niti določiti razvojnih zmogljivosti. Tak projekt bo potem razvojne narave, kjer bomo pravo rešitev šele sproti iskali (pripadnost skupini, koordinacijski mehanizem vzajemno prilagajanje, razvoj s pomočjo serije prototipov, ekstremno vodenje projektov).

21.2 Ocenjevanje stroškov

- intelektualno delo, ki ga je že sicer težko meriti
- relativna mladost programskega inženirstva in s tem povezano pomanjkanje oziroma hitro zastarevanje izkušenj
- večja je kompleksnost in velikost naloge, večja je nezanesljivost ocen
- težje je tudi ocenjevati in kontrolirati daljše projekte, saj izkušnje kažejo, da je se sodelavci pri večletnih projektih predolgo "ogrevajo", na koncu pa hitijo, saj zmanjkuje časa
- pri več let trajajočih projektih je težava tudi izredno hitro se spreminjajoča tehnologija

21.3 Merila za ocenjevanje

1. človek – mesec
   - velike razlike v produktivnosti (1 : 10)
   - razcepitveno in kompleksno delo:
2. število programskih vrstic (LOC – Lines Of Code)
3. funkcijske točke
   - točkovanje različnih notranjih struktur (število in vrsta različnih podatkovnih struktur, različnih funkcij)
   - ali zunanjih parametrov načrtovane programske opreme (število in vrsta vhodnih in izhodnih podatkovnih tipov, vmesnikov in kontrolnih struktur)

21.4 Kdo ocenjuje?

- strokovnjaki, ki podajo svoje ocene
- metoda Delphi (Koordinator od vseh ocenjevalcev zbere njihove ocene. Te ocene (brez imen njihovih avtorjev) nato razdeli med ocenjevalce. Na osnovi ocen drugih lahko vsak modificira svojo oceno. V več krogih ocene ponavadi konvergirajo v dober približek.)

- vsak ocenjevalec mora podati optimistično, realistično in pesimistično oceno

21.5 Metode za ocenjevanje

- pogosto se izhaja iz nekih zunanjih omejitev (koliko časa je maksimalno na voljo ali koliko lahko največ dobimo plačano za neko delo), potrebno je adaptivno vodenje projektov!

- tipična delitev stroškov oziroma časa med glavnimi aktivnostmi v razvojnem ciklu:

  40% analiza in načrtovanje

  20% kodiranje

  40% testiranje

Dekompozicijske metode razdelijo celotno potrebno delo na smiselne enote. Delovne enote so posamezni programski moduli (podprogrami, funkcije, vmesniki ipd.). Te manjše enote so bolj pregledne in jih zato lažje ocenimo bodisi s številom potrebnih programskih vrstic ali s funkcijskimi točkami.

S pomočjo prevzetih ali lastnih, na osnovi že končanih projektov izpeljanih meril produktivnosti (npr. [*število programskih vrstic, človek – mesec*]), ocenimo potrebni čas in zmogljivosti.

Empirične metode za določanje zmogljivosti in časa uporabljajo izkustvene formule, izpeljane na osnovi dokončanih projektov. Formule povezujejo zmogljivosti s potrebnim delom in časom. Uteži v formulah pa določimo iz tabel, kjer točke odražajo zahtevnost in funkcionalnost programov. Tudi te metode je potrebno umeriti na lastnih podatkih!

Dekompozicijske metode lahko uporabimo takrat, ko že poznamo notranjo strukturo programske opreme. Empirične metode pa lahko uporabimo le na osnovi zahtev in omejitev že bolj zgodaj v razvoju.

21.6 Empirični nelinearni model

$$E = (a + bKLOC^c) f(x_1, \ldots, x_n)$$

$KLOC$ je število programskih vrstic v tisočih
E je delo, izraženo v *človek – mesecih*
a, b in c so konstante,
$f(x_1, \ldots, x_n)$ je korekcija, ki jo določajo faktorji $x_1 \ldots, x_n$.

- osnovna enačba je izpeljana na osnovi preteklih projektov,
- osnovni faktor, ki vpliva na obseg dela, je velikost programske opreme
- korekcijski faktorji bodisi zmanjšajo ali povečajo osnovno vrednost

21.7 Kvaliteta programske opreme

- dobra kvaliteta izdelkov in storitev se izplača
- kvaliteta je nujna, da kompleksni sistemi sploh lahko delujejo
- od kvalitetne programske opreme so odvisna človeška življenja
- zaradi večje zavesti o potrebi po kvaliteti se v svetu pojavljajo novi standardi in predpisi
- skupina standardov ISO 9000 (ISO 9000–3 za razvoj programske opreme), drugi namenjeni posebej programski opremi (IEEE itd.)

21.8 Kaj je kvaliteta?

- kvaliteto je težko izmeriti
- pogosto pa je kvaliteto zelo lahko identificirati
- McCallov sistem določa tri kategorije faktorjev kvalitete programske opreme:
  - Vsak *faktor kvalitete* mora izpolnjevati določene **kriterije** (npr. popolnost, sledljivost, modularnost, samodokumentiranje, splošnost, preprostost itd.)
  - za *zanesljivost* na primer, je potrebna *popolnost, konsistentnost in sledljivost.*
  - faktorji kvalitete so lahko med seboj nasprotujoči
  - če želimo visoko učinkovitost, to slabo vpliva na možnost vzdrževanja, možnost testiranja, fleksibilnost in prenosljivost.

Delovanje produkta	
pravilnost	Ali dela, kar zahtevam?
zanesljivot	Ali deluje ves čas pravilno?
učinkovitost	Ali dela na mojem računalniku kar se da hitro?
varnost	Ali je varen?
uporabnost	Ali ga znam uporabljati?
Revizija produkta	
možnost vzdrževanja	Ali ga lahko popravim?
možnost testiranja	Ali ga lahko testiram?
fleksibilnost	Ali ga lahko spreminjam?
Tranzicija produkta	
prenosljivost	Ali ga lahko uporabljam na drugih računalnikih?
ponovna uporabljivost	Ali lahko ponovno uporabim del programske opreme?
združljivost	Ali ga lahko povežem z drugimi sistemi?

21.9 Različne definicije kvalitete

1. Transcedentalna definicija govori o notranji kvaliteti, ki jo je težko definirati, izkušeni poznavalci pa jo začutijo.

2. Uporabniška definicija govori predvsem o uporabnosti sistema in kako sistem rešuje potrebe uporabnika. Ker se uporabniki med seboj zelo razlikujejo, je nek sistem lahko za nekoga dober, za drugega pa slab. Sistem LaTeX za urejanje besedil je za povprečnega uporabnika preveč zapleten, za znanstvenika pa primeren.

3. Definicija na osnovi produkta zadeva lastnosti programske opreme. Večina sistemov za opredeljevanje kvalitete, tako kot tudi zgoraj opisani, govorijo o tej vrsti kvalitete.

4. Definicija na osnovi specifikacije opredeljuje kvaliteto glede na ujemanje s specifikacijo. To vrsto kvalitete testiramo s sistemskim testom.

5. Vrednostna definicija kvalitete se ukvarja s stroški in dobički pri načrtovanju in vodenju celotnega projekta.

21.10 Standardizacija kvalitete

Pri zagotavljanju kvalitete je možno izbrati dve poti:

- preveriti (testirati) kvaliteto gotovega izdelka,
- organizirati tak razvojni oziroma produkcijski proces, da bo izdelek zagotovo ustrezal določenim kriterijem kvalitete.

Najprej je prevladoval prvi način, sedaj pa vse bolj drugi, saj je potratno izdelati nek produkt in ga potem zavreči, če ne ustreza. Pri maloserijskih ali unikatnih produktih to še toliko bolj velja!

ISO, mednarodna organizacija za standardizacijo, je izdala številne standarde, ki govorijo o kvaliteti (serija ISO 9000).

Za razvoj programske opreme je iz te serije najbolj primeren standard ISO 9000–3

21.11 Kako izboljšati kvaliteto?

1. določiti cilje kvalitete,
2. oceniti svoj način dela in izbrati akcije za izboljšavo tega procesa,
3. med delom zbirati podatke o tem procesu ter o rezultatih oziroma produktih,
4. s pomočjo smiselnih hipotez interpretirati zbrane podatke,
5. če je potrebno, zopet korigirati proces dela.

Poglavje 22

Analiza zahtev

- podrobno identificirati in dokumentirati zahteve:
  - dejanske funkcije (po pomembnosti – MUSTs, WANTs)
  - dodatne zahteve: zmogljivost, zanesljivost, uporabnost itd.
  - platforma, uporabniki
  - povezovanje z okoljem
- zahteve so podane v posebnem dokumentu—**specifikaciji zahtev** oziroma v PDS (Project Definition Statement)
  - osnova za nadaljnje delo
  - kriterij za vrednotenje uspeha
- analizo in načrtovanje ni možno strogo ločiti
- tehnična študija izvedljivosti
- ekonomska študija

22.1 Specifikacija zahtev

IEEE poleg tega priporoča, da je specifikacija:

Enoumna — zahteve je možno le enolično interpretirati. Zaradi svoje narave je to z naravnim jezikom težko.

Popolna — vse pomembne zadeve v zvezi s funkcionalnostjo, zmogljivostjo in omejitvami naj bodo dokumentirane.

Preverljiva — pomeni, da je možno ugotoviti, če so zahteve uresničene. Fraze, kot so "sistem naj bo prijazen do uporabnika" niso objektivno preverljive.

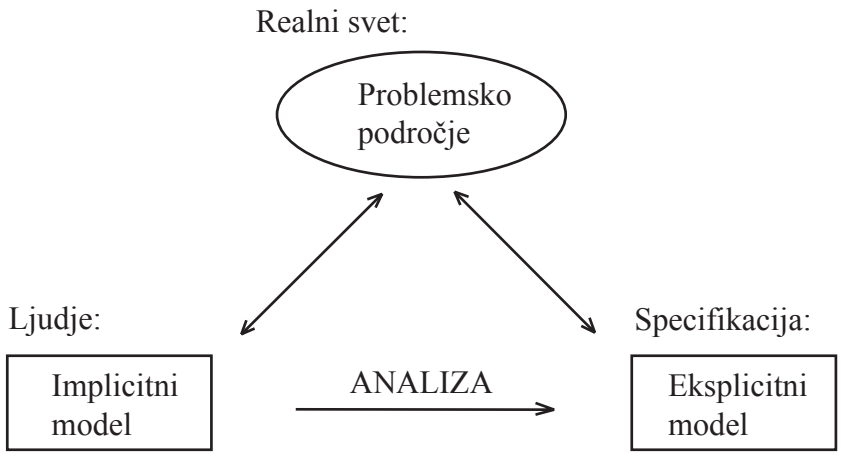
Konsistentna — zahteve si ne smejo nasprotovati niti v logičnem niti v časovnem smislu.

Spremenljiva — ker programska oprema modelira neko realnost, se mora skupaj s to realnostjo spreminjati.

Sledljiva — vzrok za vsako zahtevo mora biti natanko znan.

Uporabna — specifikacija mora biti uporabna tudi, ko sistem že deluje. Implicitno védenje, ki je igralo vlogo pri pisanju specifikacije, pa ni bilo zapisano, lahko povzroča velike težave pri vzdrževanju programske opreme.

22.2 Ljudje kot viri informacij



Slika 22.1: Ljudje si za delo na nekem problemskem področju ustvarijo implicitni model delovanja, ki ga je potrebno pri analizi ubesediti oziroma eksplicitno prikazati.

22.3 Zbiranje informacij

1. **Spraševanje:** Naročnike sprašujemo, kaj pričakujejo od novega sistema. Pri tem lahko le upamo, da zna naročnik obiti svoje omejitve in predsodke. Spraševanje lahko poteka v obliki intervjuja, "brainstorminga" ali vprašalnika.

2. **Izpeljava iz obstoječega sistema:** Analizo začnemo na osnovi obstoječega sistema, na primer podobnega sistema v neki drugi organizaciji ali opisa sistema v literaturi.

3. **Sinteza iz lastnosti okolice:** Programska oprema se bo uporabljala v določenem okolju. Da bi uspešno delovala, mora upoštevati zakonitosti okolja. Zahteve lahko zato formuliramo na osnovi analize okolja. Tovrstni analizi pravimo procesna analiza, normativna analiza ali decizijska analiza.

4. **Izdelava prototipov:** Skozi nekaj generacij prototipov, ki vodijo do vedno bolj podrobnih zahtev, pridemo do končne specifikacije. Namesto dejanskih prototipov lahko specifikacijo določimo z razvijanjem scenarija možne uporabe sistema.

22.4 Pogajalski problemi

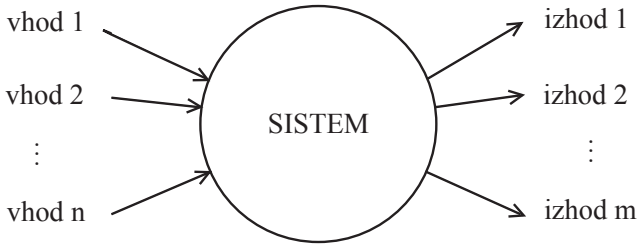
- večina strategij in analitičnih metod je v svojem bistvu tehnicističnih ali funkcionalističnih

- probleme naj bi rešili z rekurzivnim deljenjem na manjše in preprostejše probleme

- vsak problem naj bi bil rešljiv z enim najboljšim možnim načinom, ki ga lahko odkrijemo s skrbnim opazovanjem in eksperimentiranjem

- kadar so posredi ljudje analitik ne more biti le pasivni opazovalec

- nezadovoljni uporabniki sistema ne bodo uporabljali



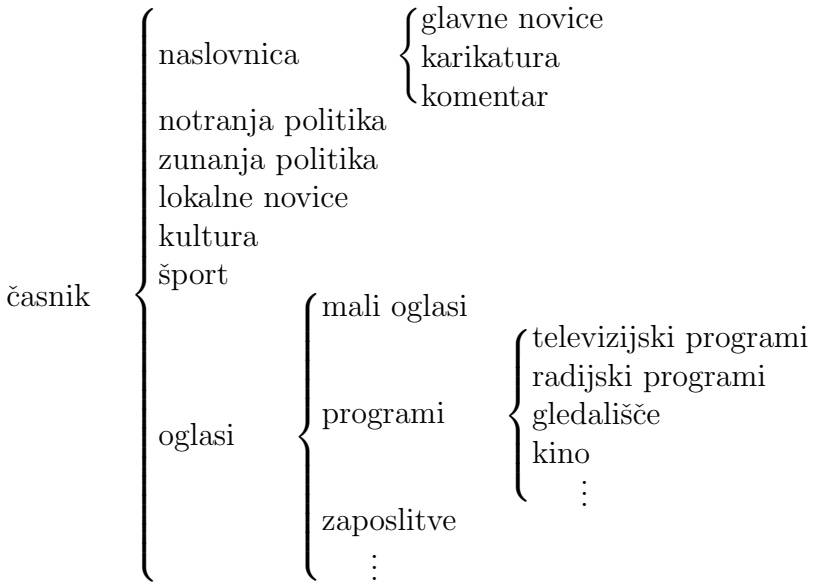
Slika 22.2: Pri analizi na osnovi podatkovnega toka je poudarek na transformaciji oziroma pretoku informacije skozi sistem. Diagram podatkovnega toka se nato pri načrtovanju preslika v programsko strukturo.

22.5 Analiza na osnovi podatkovnega toka

22.6 Analiza na osnovi strukture podatkov

22.7 Glavna načela pri analizi

- Problem moramo razdeliti na podprobleme tako, da se podrobnosti razkrijejo postopoma in na hierarhičen način.
- Specifikacija mora obsegati celoten sistem, katerega del je programska oprema, in opisati vse okoliščine, v katerih sistem deluje.
- Ločiti moramo funkcije od njihove implementacije. Odgovoriti moramo na vprašanje, **KAJ** je potrebno, preden odgovorimo na vprašanje **KAKO** to narediti. Specifikacija je model za *razumevanje*, ne pa *načrt* za implementacijo.
- Na osnovi specifikacije naj bo moč ugotoviti, če nek sistem izpolnjuje zahteve.
- Specifikacija naj bo modularna. Če pride do sprememb, naj bo potrebno spremeniti čim manjši del specifikacije.



Slika 22.3: Pri analizi na osnovi strukture podatkov je poudarek na hierarhični strukturi podatkov, ki se nato pri načrtovanju preslika v programsko strukturo.

Poglavje 23

Načrtovanje

specifikacija → **načrt programske opreme** → kodiranje

- kvaliteta programske opreme je v največji meri odvisna od načrtovanja
- slabo načrtovana ali programska oprema, ki je nastala brez načrta, odpove že pri majhnih spremembah, jo je težko testirati in vzdrževati
- načrt naj ima hierarhično in modularno strukturo
- časovno ločimo preliminarno in podrobno načrtovanje
- s tehničnega vidika ima načrt tri sestavine:
  1. načrt programske arhitekture,
  2. načrt podatkovnih struktur,
  3. načrt posameznih procedur (modulov).

23.1 Arhitektura programske opreme

- arhitektura določa nadrejenost oziroma podrejenost posameznih modulov
- ne kaže pa samega vrstnega reda procesiranja, odločitev ali ponavljanj operacij

23.2 Načrt podatkovnih struktur

- podatkovne strukture ponazarjajo logična razmerja med posameznimi podatkovnimi elementi
- struktura podatkov vpliva na organizacijo, način dostopa in povezave med njenimi elementi ter s tem na sam načrt postopkov procesiranja
- v postopku načrtovanja je pomembno, da poiščemo oziroma načrtujemo ustrezne podatkovne strukture
- poleg klasičnih podatkovnih struktur (vektorji, skladi, vrste, drevesa itd.), ki so v izbranih programskih jezikih lahko celo neposredno vgrajene, je pomembna zmožnost definiranja novih podatkovnih struktur

23.3 Načrt postopkov

- določiti moramo postopke procesiranja v posameznih modulih
- strukturirano programiranje je izpostavilo tri osnovne elemente procesiranja: *zaporedje*, *pogoj* in *ponavljanje*, ki omogoča opis kateregakoli algoritma
- le v izjemnih primerih je smiselno uporabiti ukaz *GO TO*.
- potrebni so formalni zapisi, kot so diagrami poteka, psevdokoda in tabelarni zapisi

23.4 Načrtovalska načela

Postopna izboljšava

- postopno razkrivanja podrobnosti od zgoraj navzdol
- na vsakem koraku izboljšave moramo funkcijo oziroma ukaz nadomestili z bolj podrobnimi ukazi
- primerno predvsem za načrtovanje majhnih programov

Abstrakcija

- zaradi kompleksnosti je naenkrat nemogoče obravnavati in razumeti vse podrobnosti
- na najvišjih ravneh abstrakcije govorimo o rešitvah v obliki nalog, ki jih mora programska oprema izvajati
- na nižjih ravneh abstrakcije pa za posamezne dele programske opreme že snujemo postopke za izvajanje posameznih funkcij
- na najnižji ravni izdelamo načrt, ki je neposredno uporaben za kodiranje

Postopkovna abstrakcija

- da bi rešili nek zapleten problem, razmišljamo po kakšnih korakih priti do cilja
- posamezni koraki na poti rešujejo podprobleme, katerih rešitve sestavljajo celotno rešitev
- na najnižji ravni abstrakcije, so preprosti problemi, ki so enostavno rešljivi v izbranem jeziku
- večina programskih jezikov nudi kontrolne strukture, ki omogočajo postopkovno reševanje problemov
- primer je vhodno-izhodno procesiranje, ko si posamezni moduli podajajo podatke
- tako zasnovane programe je težko spreminjati in prilagajati, saj morajo biti moduli med seboj zelo usklajeni, predvsem glede strukture podatkov

Podatkovna abstrakcija

- omogoča razgraditev problemov na način, ki pripelje do bolj neodvisnih modulov
- podatkovna abstrakcija išče hierarhijo v podatkih
- iz preprostih podatkovnih struktur je možno zgraditi bolj kompleksne, abstraktne podatkovne strukture, ki so prilagojene podatkom problema, ki ga rešujemo
- namesto, da bi se za določeno obliko predstavitve podatkov odločili že na najvišji ravni obravnave problema in to rešitev vsilili oziroma razkrili vsem modulom, podrobnosti o dejanski podatkovni strukturi raje skrijemo v poseben modul

- novejši programski jeziki (npr. Ada, Modula-2, Oberon, C++) omogočajo poleg postopkovne tudi podatkovno abstrakcijo
- načelo podatkovne abstrakcije vodi do objektno orientiranega načrtovanja

Modularnost

- modularnost omogoča razumevanje kompleksnih sistemov, saj naenkrat ne moremo imeti v glavi velikega števila spremenjivk, odločitev in različnih poti skozi programsko kodo
- merilo modularnosti je:
  - notranja enostnost modulov
  - soodvisnost modulov

Notranjo enotnost načrtovanega modula lahko enostavno preizkusimo tako, da namen modula opišemo z enim stavkom:

- če ima stavek več kot en glagol ali nekaj našteva, modul opravlja več kot eno samo funkcijo
- če so v stavku besede, ki označujejo vrstni red ali časovno zaporedje, ima modul časovno ali zaporedno notranjo enotnost

Soodvisnost (angl. coupling) je mera povezanosti med moduli, večja ko je soodvisnost med moduli, slabša je programska struktura.

23.5 Načrtovanje modulov

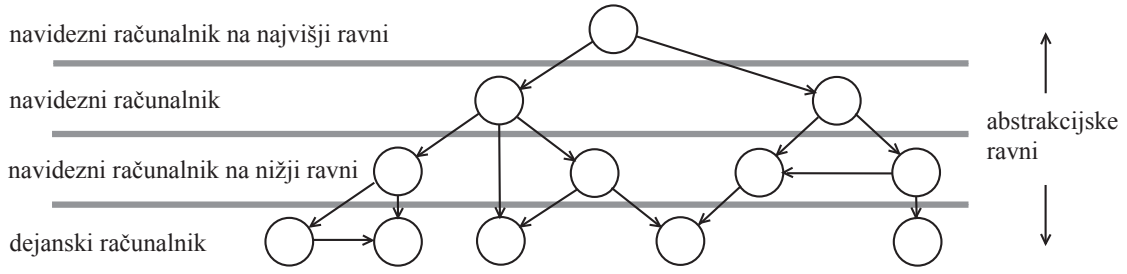
- veliko notranjo enotnost in majhno soodvisnost dosežemo predvsem tako, da modulom načrtujemo čim bolj preproste vmesnike
- med moduli naj bi prišlo do izmenjave informacij le kot posledica ustreznega klica:
  - sodelovanje med programerji je lažje,
  - manj verjetno je, da bi spremembe v enem modulu vplivale na druge,
  - lažje je ponovno uporabiti module v drugih sistemih,
  - delovanje modulov je lažje razumeti,

  - izkušnje kažejo, da je v takih modulih manj napak.
- čeprav je programski sistem načrtovan modularno, se včasih lahko zaradi hitrosti ali omejenega spomina zakodira na "monolitni" način (npr. programska oprema za mikroprocesorje)

23.6 Načrtovalske metode

- metoda načrtovanja sestoji iz množice navodil, načrtovalskih izkušenj in postopkov
- za vsako metodo je značilna tudi notacija, s katero izrazimo rezultat načrtovanja
- obstaja cela vrsta načrtovalskih metod
- nekatere metode predpisujejo predvsem notacijo, druge pa tudi natančno določajo sam postopek načrtovanja
- številne metode podpirajo tudi posebna orodja CASE.
- smiselno je, da je metoda načrtovanja usklajena z metodo analize, na primer na osnovi pretoka podatkov ali na osnovi strukture podatkov

23.7 Funkcijska dekompozicija



- funkcijo, ki jo želimo implementirati, razdelimo na podfunkcije, ki vsaka rešuje del problema
- vsako podfunkcijo lahko zopet delimo naprej po načelu "deli in vladaj" dokler ne pridemo do osnovnih funkcij in struktur izbranega programskega jezika

- za povezovanje teh dveh koncev obstajata dva osnovna načina:

  Načrtovanje od zgoraj navzdol začne z delitvijo glavnih funkcij programske opreme in to delitev ponavlja. Tak način je možen le, če so vse glavne funkcije sistema natančno določene.

  Načrtovanje od spodaj navzgor iz osnovnih gradnikov s pomočjo abstrakcije postopoma gradi glavne funkcije. Ta način je sicer bolj prilagodljiv, vendar lahko svoj cilj (sistemske funkcije) zgreši.

- načrtovanje je dejansko bolj podobno igrači "yo-yo" — načrtovalci se po potrebi in svojem nagibu sprehajamo med različnimi abstrakcijskimi ravnmi

23.8 Izbiranje načrtovalske metode

- nobena metoda ne daje popolnega recepta, kako iz spiska zahtev zgraditi kvaliteten programski produkt
- izkušnje razvijalcev imajo odločilen vpliv pri vsakem razvoju programske opreme
- nekatere metode načrtovanja programske opreme bolj natančno določajo postopek reševanja, druge so zelo splošne
- kadar je potrebno definirati strukturo podatkov, so primerne metode na osnovi strukture podatkov in objektno orientirane metode
- metode, ki temeljijo na pretoku podatkov, so primerne predvsem za avtomatizacijo obstoječih sistemov
- zaradi vzdrževanja mora biti programska oprema fleksibilna, razumljiva in modularna, kar je tudi pomemben kriterij izbire načrtovalske metode
- na izbiro pa vplivajo tudi naslednji faktorji:
  - izkušnje na uporabniškem področju,
  - izkušnje z načrtovalsko metodo,
  - razpoložljiva orodja,
  - celostna filozofija pristopa k razvoju programske opreme.

Poglavje 24

Kodiranje

- če je načrt skrbno in dovolj podrobno pripravljen, je kodiranje pretežno mehansko opravilo
- od izbire programskega jezika in načina kodiranja je kvaliteta programske opreme neposredno odvisna
- pri izbiri programskega jezika igra vlogo več faktorjev:
  - največji vpliv na izbiro naj ima narava sistema
  - za reševanje določenega problema so primerni jeziki, s katerimi je lažje izraziti kontrolne in podatkovne strukture in druge zahteve problemskega področja
  - programerji izbirajo programski jezik tudi glede na njihovo seznanjenost z jeziki in izkušnje
- psihološke lastnosti programskih jezikov lahko povečajo možnost napak med programiranjem
- tehniške lastnosti

24.1 Psihološke lastnosti programskih jezikov

- vplivajo na sam proces kodiranja kot človeške aktivnosti
- odločajo o naši sposobnosti učenja in uporabe programskega jezika ter vzdrževanja programske kode

- naše razmišljanje o rešitvah je lahko nehote omejeno z lastnostmi izbranega programskega jezika
- *Semanatično znanje* obsega splošne koncepte, ki niso neposredno vezani na določen programski jezik.
- *Sintaktično znanje* obsega pravila, kako se pišejo na primer zanke, prireja vrednosti spremenljivkam, kako se posega v podatkovne strukture itd.
- Na osnovi poznavanja semantičnih struktur se lahko hitro naučimo novega programskega jezika

Uniformnost. Stopnja konsistentne uporabe notacije je odvisna od uniformnosti programskega jezika (krogle oklepaje uporabljamo v aritmetičnih izrazih, za omejitev števcev v vektorskih podatkovnih strukturah in omejitev argumentov podprogramskih klicev). Zaradi pomanjkanja simbolov v naboru znakov ASCII isti znaki označujejo različne operacije (ang. operator overloading).

Dvoumnost. Prevajalnik vsak sintaktično pravilno zapisan izraz interpretira na enak način. Če prevajalnik določa vrstni red operacij glede na vrstni red zapisa, na primer od leve proti desni, bo izraz `x = a/b*c` interpretiral kot `x = (a/b) * c`. Vendar lahko isti izraz nek programer razume tudi kot `x = a/(b * c)`.

Pogost vir dvoumnosti so tudi implicitne deklaracije podatkovnega tipa spremenljivk.

Kompaktnost programskega jezika je odvisna od vrste in števila sintaktičnih informacij, ki si jih moramo zapomniti. Na kompaktnost programskega jezika vpliva:

- število in raznolikost kontrolnih in podatkovnih struktur
- vrsta besed (njihova dolžina!) in okrajšave, ki jih lahko uporabljamo
- število vgrajenih funkcij in operacij

24.2 Tehnične lastnosti

Tehnične lastnosti programskih jezikov odločajo predvsem o primernosti programskega jezika za določeno vrsto problema. Osnovne lastnosti programskih jezikov, gledano s tehničnega vidika, so:

1. težavnost prevajanja načrta programske opreme v kodo,
2. učinkovitost prevajalnika,
3. prenosljivost kode,
4. ustrezna programska razvojna orodja,
5. zmožnost vzdrževanja.

24.3 Specializacija programske kode

- pri kodiranju naj bi stremeli k čim bolj enostavnim in splošnim rešitvam
- taka koda je lažje prenosljiva in omogoča lažje popravljanje ter testiranje
- koda naj bo zato le tako hitra oziroma učinkovita, kot to zahteva specifikacija in naj **ne** bo najhitrejša (najboljša) možna!
- za večjo hitrost izvajanja programa, moramo včasih določen del programske kode specializirati
  - poenostavitev aritmetičnih in logičnih izrazov
  - optimizacija zank
  - uporaba enostavnih podatkovnih struktur, celoštevilčnih ali Boolovih spremenljivk
  - obravnava izjemnih primerov, ki hitro rešijo del primerov ali posebej pogoste primere
  - dele sistema zakodirati v zbirnem jeziku ali celo uporabiti posebno strojno opremo

24.4 Dokumentiranje programske kode

- komentarji med kodo so za hitro razumevanje kode neobhodni
- pomembno je sprotno dokumentiranje kode
- interna dokumentacija programske kode sestoji iz uvodnih komentarjev in komentarjev med kodo, važna pa je tudi logična in vizualna organizacija kode

- *komentarji med kodo* naj bodo le tam, kjer je kakšna posebnost
- za berljivost in razumljivost kode poskrbimo v prvi vrsti z ustrezno izbiro mnemoničnih imen

```
    d  =  v * t
 dist  =  vel * time
dist_m =  vel_m_per_sec * time_in_sec
```

- *vizualno organizacijo* dosežemo z zamikanjem začetka vrstic, izpuščanjem praznih vrstic itd.
- *kratki testni primeri*

Poglavje 25

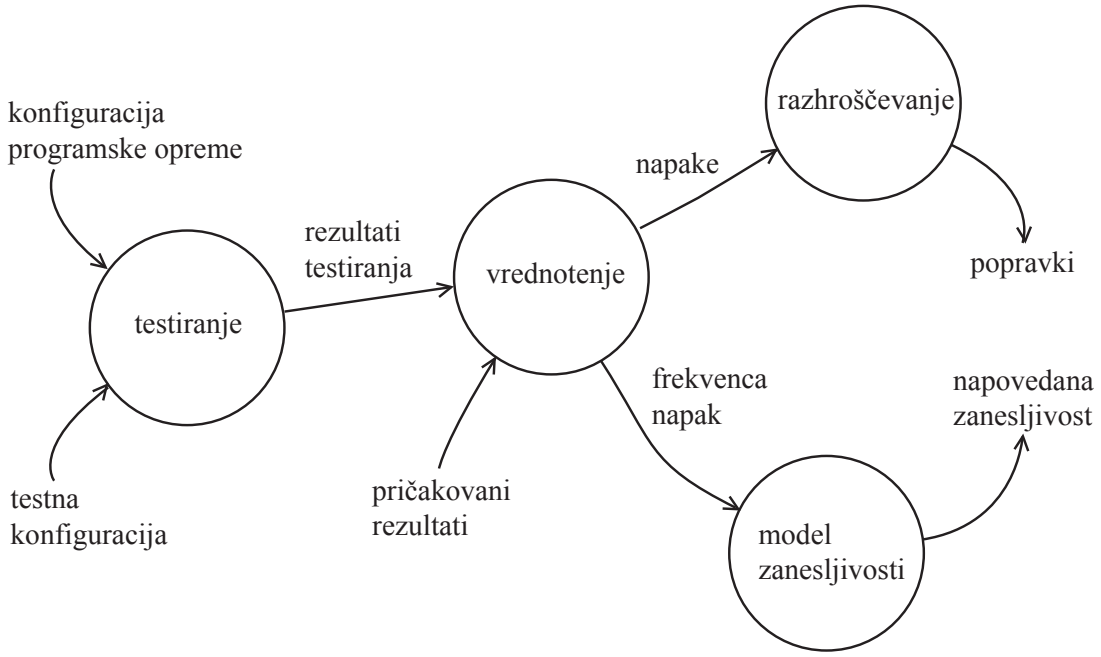
Testiranje

- testiranje je osrednja aktivnost, ki preverja kvaliteto programske opreme
- s primernim nadzorom razvoja programske opreme želimo napake že vnaprej *preprečevati* in pomanjkljivosti *sproti* odkrivati
- kasneje v toku razvoja programske opreme kaj spremenimo, dražja je sprememba
- testiranje je integralni del nadzora kvalitete celotnega razvojnega cikla
- rezultat vsake razvojne faze kvaliteten in v skladu z globalnimi cilji

25.1 Verifikacija in validacija

1. Ali programsko opremo gradimo pravilno? **Verifikacija** ugotavlja predvsem tehnično pravilnost zadnje faze razvoja.
2. Ali programska oprema ustreza osnovnim zahtevam? **Validacija** preverja, če gradimo pravi produkt.

- testiranje je s psihološkega vidika destruktivno delo (konflikt interesov)
- testiranje naj preverja kodo, ne pa razvijalce!



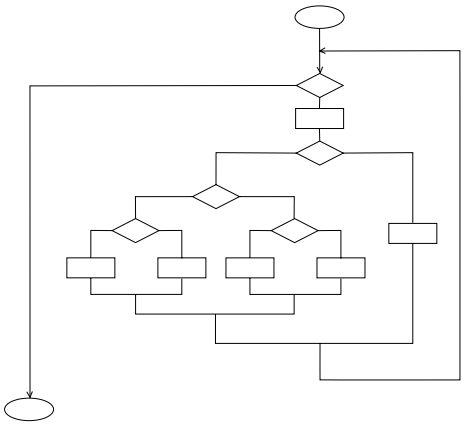
Slika 25.1: Potek testiranja

25.2 Zakaj popolno testiranje ni možno?

- testiranje programa lahko pokaže samo na prisotnost napak, ne more pa dokazati, da napak v programu ni!
- popolno testiranje pomeni izvajanje programa po vseh možnih poteh skozi program
- število različnih možnih načinov izvajanja programa eksponentno narašča
- če bi se zanka v programu, ki ga prikazuje spodnji diagram poteka ponovila dvajsetkrat, bi bilo število možnih načinov izvajanja programa okoli:

$$N = 5^{20}.$$

- če za en testni primer potrebujemo le stotinko sekunde, bi to trajalo več kot 30.000 let



25.3 Kaj je potem cilj testiranja?

- povečati naše zaupanje v programsko opremo
- na osnovi izkušenj so se izoblikovale metode testiranja, ki preverjajo delovanje programske opreme na najbolj kritičnih mestih in na mestih, kjer se napake najpogosteje pojavljajo
- preverimo delovanje vsake posamezne instrukcije v programski kodi
- obnašanje celotnega sistema skušamo preveriti v vseh možnih realnih pogojih delovanja
- upamo lahko, da bomo odkrili vsaj večino napak

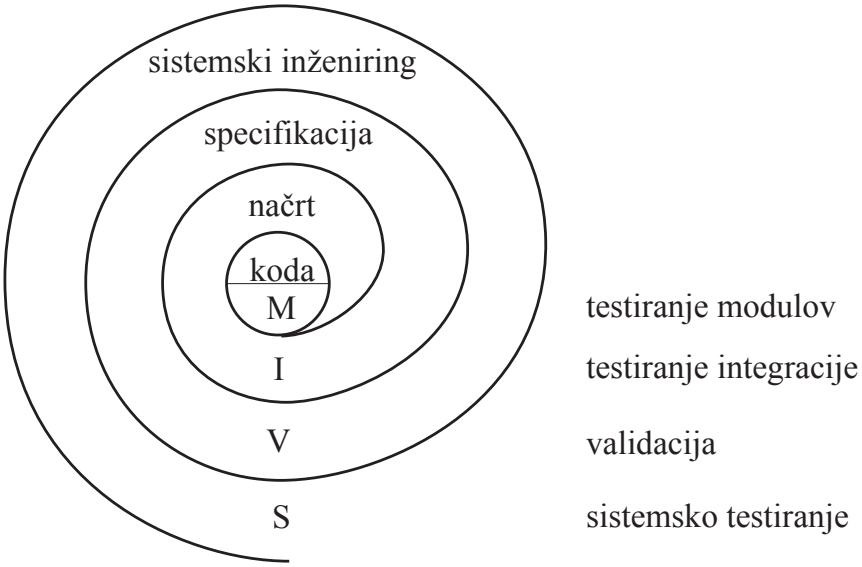
25.4 Statične metode testiranja

Statične metode, ki ne zahtevajo izvajanja programa

Branje. Zaradi tega avtor programa postane skoraj slep za nekatere pomanjkljivosti ali napake, saj bolj misli na to, kako naj program dela. Boljše rezultate dosežemo, če kodo bere kdo drug kot in ne njen avtor.

Najbolj neodvisno oceno dobimo z anonimno recenzijo (angl. peer review).

Sledenje in preverjanje. Preverjanje programske opreme izvajamo v obliki formalnih recenzijskih sestankov. Na sestanku se kodo prebira in hkrati razlaga njen pomen. Ta interpretacija kode lahko pomaga pri odkrivanju napak.



Slika 25.2: Strategija testiranja

Sledenje (angl. walktrough) je preverjanje programske kode s pomočjo testnih podatkov. Ker je metoda ročna, morajo biti testni primeri preprosti, saj postane sledenje hitro prezahtevno.

Postopna abstrakcija. Iz programske kode skušamo v nekaj korakih razbrati funkcijo, ki jo ta koda implementira. Funkcijo, ki jo na ta način abstrahiramo, primerjamo s funkcijo, opisano v specifikaciji ali načrtu za ta del programske kode.

25.5 Dinamične metode testiranja

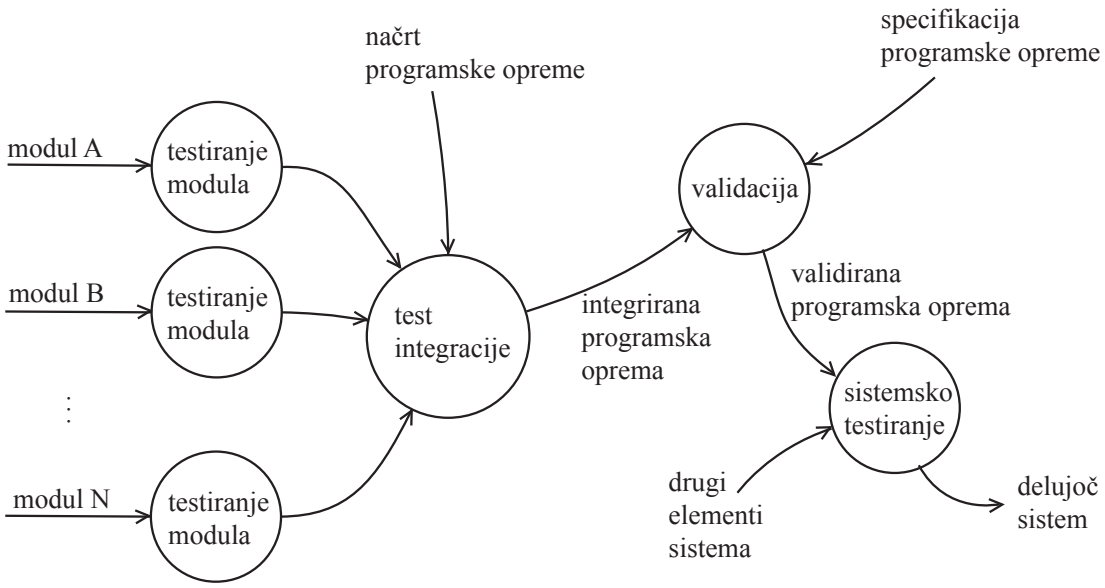
Metode bele skrinjice ali **strukturna analiza**. Metode v tej skupini pri načrtovanju testnih primerov upoštevajo notranjo strukturo kode.

- po principu bele skrinjice testiramo predvsem posamezne module
- med metode bele skrinjice sodita:
  1. *Testiranje glavnih poti.* Osnovni namen metode je, da se vsi programski ukazi izvedejo vsaj enkrat.
  2. *Testiranje zank.*

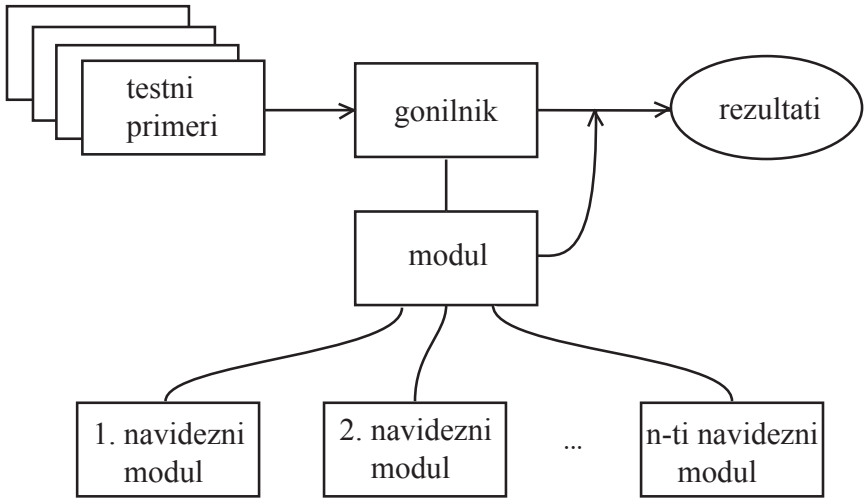
Metode črne skrinjice ali **funkcionalna analiza**. Pri teh metodah testiranja je notranja struktura kode zastrta. Izhodne rezultate primerjamo s pričakovanimi.

- uporabljamo jo v kasnejših fazah testiranja
  - napačne ali manjkajoče funkcije,
  - napake vmesnika,
  - napake pri branju podatkov v podatkovnih bazah,
  - nizka zmogljivost,
  - napake pri inicializaciji in na koncu procesiranja.
- s čim manjšim številom testov želimo odkriti čimveč možnih napak (ekvivalentne particije, analiza mejnih vrednosti)

25.6 Vrstni red testiranja



Slika 25.3: Vrstni red testiranja



Slika 25.4: Testiranje modulov

25.7 Testiranje integracije

- postopno, regresijsko testiranje
- vrstni red integracije

  Integracija od zgoraj navzdol. Začnemo z glavnim ali kontrolnim modulom. Postopoma dodajamo podrejene module, v globino ali v širino. Če integriramo najprej v globino, lahko prej demonstriramo določeno funkcijo sistema. Potrebni so le začasni podrejeni moduli.

  Integracija od spodaj navzgor poteka tako, da module na nižjem nivoju združimo v skupine. Napisati moramo gonilnike, ki koordinirajo testiranje skupin modulov. Postopoma odstranjujemo gonilnike, jih nadomeščamo s pravimi moduli ter skupine združujemo. Podobno kot pri navideznih modulih tudi pri gonilnikih lahko po kompleksnosti ločimo štiri vrste gonilnikov.

25.8 Sistemsko testiranje

- nastopi, ko je sistem že integriran
- hitrost programske opreme

- zmožnost okrevanja sistema po izpadu sistema
- maksimalna obremenitev sistema
- varnost
- občutljivost
- α in β verzija

Poglavje 26

Vzdrževanje programske opreme

1. **Popravki za odpravljanje napak.**

2. **Prilagajanje programske opreme** spremembam v okolju, na drugo vrsto strojne ali sistemske programske opreme. Pri tem se funkcionalnost programske opreme ne spremeni.

3. **Razširitev zmogljivosti programske opreme** se ukvarja z novimi ali spremenjenimi zahtevami. Spremeniti moramo funkcije sistema, izboljšamo lahko zmogljivost sistema ali uporabniški vmesnik.

4. **Preventivno vzdrževanje programske opreme** je namenjeno spremembam, ki izboljšajo možnost vzdrževanja (dopolnjevanje dokumentacije, izboljšava programske strukture itd.).

26.1 Glavni problemi pri vzdrževanju

- nepopolna specifikacija in slaba dokumentacija

- ni testnih primerov

- nepriljubljeno delo

- spremembam se ni možno izogniti, saj je programska oprema del nenehno se spreminjajoče realnosti

26.2 Organizacija vzdrževanja programske opreme

Skupine za razvoj in vzdrževanje programske opreme lahko organiziramo po treh načelih:

W-type: delitev po tipu dela (angl. work), na primer analiza in programiranje.

A-type: delitev glede na vrsto aplikacije.

L-type: delitev glede na življenjski cikel (angl. life-cycle), na primer razvoj programske opreme in vzdrževanje programske opreme.

26.3 Ločitev razvoja in vzdrževanja

Prednosti ločenega razvoja in vzdrževanja so:

- jasne zadolžitve; če isti ljudje razvijajo in vzdržujejo programsko opremo, je težko nadzorovati, koliko časa kaj delajo;
- zaradi nujnih vzdrževalnih del je težko časovno načrtovati razvoj nove programske opreme;
- delitev razvoja in vzdrževanja postavlja jasno ločnico med obe aktivnosti; s to ločnico so običajno povezani zaključni testi;
- s specializacijo aktivnosti je možna višja kvaliteta;
- z bolj osredotočenim delom je možna večja produktivnost.

Slabosti ločenega razvoja in vzdrževanja programske opreme so:

- slabša motivacija ljudi zaradi različnega statusa dela, ki ga opravljajo;
- slabše poznavanje sistema (načrta in aplikacije);
- slabša koordinacija med razvojem in vzdrževanjem, še posebej ko gre za razvoj novega sistema, ki bo nadomestil starega;
- večji začetni stroški vzdrževanja;
- možno podvajanje komunikacij z uporabnikom.

26.4 Postopek vzdrževanja

1. Vzdrževanje programske opreme moramo ustrezno organizirati, da ne bi prišlo do neavtoriziranih ali nasprotujočih sprememb programske opreme.

2. Vsak zahtevek za vzdrževanje mora biti skrbno dokumentiran. V primeru napake moramo dokumentirati okoliščine, ki so pripeljale do napake.

3. Zahtevke moramo preveriti in odobriti le tiste, ki so potrebni, smiselni in ekonomsko upravičeni.

4. Glede na vrsto vzdrževanja sprožimo ustrezen postopek. V primeru kritične napake, ki zaustavi normalno delovanje neke organizacije, moramo takoj ukrepati z vsemi razpoložljivimi silami. Če gre za zadevo, ki še lahko počaka, jo uvrstimo na seznam napak, ki jih rešujemo po vrsti na ustaljen način.

5. Z organizacijskega vidika obravnavamo vsak zahtevek podobno kot razvoj nove programske opreme. Zahtevo za vzdrževanje moramo skrbno analizirati, spremembo ali popravek načrtovati in na koncu programsko opremo testirati.

6. Vse spremembe moramo dokumentirati. Tudi vso ostalo dokumentacijo, na primer navodila za uporabo, moramo ustrezno popraviti.

Poglavje 27

Dojemanje časa in sodobna informacijska tehnologija

- Notranji čas (kako čas dojemamo) vs. koncept časa (čas ure)
- včasih čas kar leti, drugič se čas neznosno vleče
- dojemanje časa je odvisno od zunanjih impulzov, vendar imamo tudi notranjo uro
- poskusi, ko so se ljudje povsem izolirali, dan se jim je podaljšal na ca. 25 ur
- leta 2008 so ljudje konzumirali trikrat več informacij na dan kot leta 1960
- uporabniki računalnikov med delom preklapljajo med okni ali preverjajo email 37-krat na uro
- to je ena od največjih razlik v človeškem okolju
- ljudje so se tako navadili na spremembo pozornosti in na opravljanje več opravil hkrati, da imajo težavo, da se skoncentrirajo za dalj časa na eno samo stvar
- ali se možgani pri mladi generaciji drugače razvijejo zaradi takšne stimulacije?
- raziskave so pokazale, da multitaskerji pravzaprav potrebujejo več časa, da preklopijo med nalogami in da raje iščejo nove informacije namesto da bi uporabili stare bolj uporabne informacije
- multitaskerji so bolj občutljivi na vhodne informacije

- tehnologija intenzivira star konflikt v možganih: del možganov deluje kot kontrolni center, ki omogoča, da se koncentriramo in določamo prioritete
- bolj primitivni deli možganov, ki procesirajo vidne in slušne signale, pa zahtevajo, da možgani tem informacijam prisluhnejo
- to je imelo razvojno funkcijo, saj če je v bližini zarjovel lev, je bilo smiselno prekiniti z gradnjo kolibe
- danes na podoben način znak ob prispetju novega emaila prekine naše delo npr. na domači nalogi
- vedno več ljudi že ob najmanjšem dražljaju pričakuje, da se bo nekaj zanimivega zgodilo
- naše multitasking tendence enostavno ne moremo več izključiti
- po drugi strani raziskave kažejo, da uporabniki interneta kažejo večjo možgansko aktivnost
- debata o tem, kako informacijska tehnologija vpliva na naše možgane je zelo živahna
- mediji nas spreminjajo, neustavljiva želja, da kontroliramo npr. email na naših pametnih telefonih
- negativni vpliv pretirane odvisnosti od tehnologije staršev se kaže tudi pri vzgoji otrok, ker jim starši ne dajejo dovolj pozornosti
- nove mobilne naprave omogočajo, da vsak prosti trenutek "produktivno" izrabimo
- toda če so naši možgani stalno vzburjeni in nimajo časa "počivati" to slabo vpliva na zmožnost učenja in pomnenja
- navidezen počitek možganom omogoča, da sprocesirajo informacije in dogodke, da se bolje shranijo v dolgoročni spomin
- če so možgani stalno stimulirani, učenje oziroma pomnenje slabše deluje
- ljudje mislijo, da se sprostijo, vendar se njihovi možgani utrujajo
- večina računalniški iger na mobilnih napravah se igrajo v časovnih segmentih okoli 2 minute

Poglavje 28

Upravljanje s časom po Brianu Tracyju — pojej živo žabo!

- ključ večje storilnosti je , da vsako jutro najprej opravite najpomembnejšo nalogo, da pojeste svojo žabo preden storite kar koli drugega in to brez odvečnega razmišljanja in zavljačevanja
- ljudje zamenjujejo dejavnosti z dosežki
- uspeh v življenju je dosti bolj odvisen od navad kot od naših talentov
- po uspešno opravljeni nalogi se v naših možganih sprostijo endorfini, ki povzročajo večja jasnost, samozavest in odvisnost
- z vajo pozitivna načela postanejo del nas

1. Pripravite mizo!

   natančno določite, kaj bi radi. Jasnost je bistvena. Preden začnete z delom, si zapišite cilje.

   - preden lahko kaj naredimo, moramo vedeti, KAJ želimo doseči?
   - glavni razlog za odlašanje in pomanjkanje motivacije, je zmeda in nejasnost glede naših ciljev
   - le 3% odraslih ima jasne zapisane cilje, ti ljudje dosežejo pet do desetkrat več od ljudi z enako izobrazbo in sposobnostmi, ki pa si nikoli niso vzeli dovolj časa, da bi natančno zapisali, kaj hočejo!
   - zapišimo cilje na papir: Formula za doseganje ciljev

(a) Natančno določite, kaj želite
(b) To zapišite
(c) Cilju določite rok
(d) Napravite seznam vseh stvari, ki jih boste morali storiti, da bi svoj cilj dosegli
(e) Iz seznama napravite načrt
(f) Na osnovi načrta ukrepajte TAKOJ
(g) Vsak dan storite nekaj, kar vas bo popeljalo bližje k vašemu cilju

2. Vsak dan načrtujte naprej!

   Razmišljajte na papirju. Vsaka minuta, ki jo namenite načrtovanju, vam lahko prihrani pet do deset minut pri dejanskem delu.

   - napravite seznam, kaj želite napraviti za vsak dan
   - če seznam napravimo že prejšnji večer, se lahko naša podzavest z njim ukvarja vso noč
   - različni seznami: splošni, mesečni, tedenski, dnevni
   - razvrščanje nalog po prioriteti

3. Uporabite pravilo 80/20

   dvajset odstotkov vaših dejavnosti prinaša 80 odstotkov rezultatov. Vedno se osredotočite na teh zgornih 20 odstotkov.

   - Paretovo načelo: 20% naših dejanj pripomore k 80% rezultatov
   - od 10 stvari na seznamu, sta dve pet do desetkrat pomembnejši od ostalih
   - večina ljudi pa se ukvarja z manj pomembnimi osmimi nalogami
   - najpomembnejše naloge so ponavadi najtežje in najbolj zapletene, a vendar je nagrada za njih lahko izjemna
   - dokler ne opravite 20% najpomembnejših nalog, se ne lotite ostalih 80%
   - uprite se skušnjavi, da bi opravili najprej manj pomembne naloge

4. Upoštevajte posledice

   vaše najpomembnejše naloge in proritete so tiste, ki lahko imajo najbolj resne posledice, pozitivne ali negativne, na vaše življenje in delo. Osredotočite se predvsem na te.

- razmišljanje na dolgi rok je za uspeh bolj pomembno kot družinsko ozadje, izobrazba, inteligenca in poznanstva
- dolgoročno razmišljanje izboljša kratkoročne odločitve
- razmišljanje o možnih posledicah svojih izbir, odločitev in vedenja, omogoča določitev svojih prioritet pri delu in v osebnem življenju

5. Ravnajte se po metodi ABCDE

   preden se lotite dela po seznamu opravil, jih uredite po vrednosti in prioritetah. Tako se boste resnično ukvarjali s svojimi najpomembnejšimi nalogami.

   - postavljanje prioritet:
     (a) naloga, ki jo morate opraviti, če več potem a1, a2, ...
     (b) naloga, ki bi jo morali narediti, ne delaj b-jev, dokler ne napravis a-jev
     (c) naloga, ki bi jo bilo dobro opraviti, vendar ne bo posledic, če jo ne
     (d) naloga, ki jo lahko poverite drugemu
     (e) naloga, ki jo lahko zbrišete s seznama

6. Osredotočite se na ključna področja rezultatov

   ugotovite, katera so področja, na katerih morate svoje delo brezpogojno opraviti brezhibno.

   - Zakaj dobivamo plačo? Za določene rezultate: naloge, za katere smo odgovorni, ki jih ne bo opravil nihče namesto nas
   - naredi seznam ključnih področij rezultatov in jih oceni z 1–10, kje smo šibki?
   - naše najšibkejše področje rezultatov določa nivo, do katerega bomo uporabili vse druge naše spretnosti in sposobnosti
   - katera spretnost, če bi jo razvili do popolnosti, bi najbolj pozitivno vplivala na našo kariero?

7. Ubogajte zakon prisiljene učinkovitosti

   nikoli ni dovolj časa, da bi postorili vse, vedno pa je dovolj časa za pomembn stvari. Katere so?

   - Nikoli ni dovolj časa, da bi opravili vse, kar moramo

- kup nalog ne bo nikoli prazen
- pritisk rokov ne povečuje storilnost ampak le povečuje stres
- zato: katere so moje najpomembnejše dejavnosti?
- Kaj lahko storim izključno jaz in nihče drug kot jaz, da bi – če bi to stvar opravil – ta pomenila resnično spremembo?
- Kako lahko svoj čas ta trenutek najkoristneje porabim?
- Goethe: Najpomembnevše stvari ne smejo biti podrejene najmanj pomembnim stvarem!

8. Preden začnete, se temeljito pripravite

   Pravilna predhodna priprava prepreči slabo opravljeno delo.

   - počistite svojo delovno površino
   - delovno okolje si uredite tako, da boste zmogli pri delu vztrajati dalj časa
   - pripravite vse pripomočke, gradiva, knjige, zvezke

9. Naredite svojo domačo nalogo

   Več znanja in izkušenj pri opravljanju ključnih nalog si pridobite, hitreje se jih boste lotili in prej jih boste opravili.

   - izboljšuj se na vseh ključnih področjih delovnih nalog
   - stalno učenje in izpopolnjevanje
   - ne dovoli, da bi ti šibkost in pomanjkanje znanja oviralo (slepo tipkanje, poslušanje izobraževalnih vsebin v avtomobilu)

10. Izkoristite svoje posebne talente

    Natančno ugotovite, pri katerih stvareh ste najboljši, oziroma bi lahko bili najboljši. Popolnoma se posvetite temu, da bi te ključne stvari opravljali zelo zelo dobro.

    - Pri katerih stvareh sem res dober?
    - Kaj mi pri delu nudi največ zadovoljstva?
    - Kaj bi delal, če bi zadel na loteriji?

11. Ugotovite, katere so vaše ključne omejitve

    določite svoja ozka grla, notranja ali zunanja, ki postavljajo hitrost, s katero dosegate svoje najpomembnejše cilje

12. En korak naenkrat

    korak za korakom lahko opravite tudi najobsežnejšo in zapleteno nalogo

13. Sprejmite pritisk

    zamislite si, da bi morali za en mesec odpotovati in delajte tako, kot da bi morali pred odhodom dokončati vse pomembne naloge

14. Povečajte svojo osebno moč

    ugotovite, kdaj v dnevu imate največ umske in telesne energije. Delajte takrat! Privoščite si veliko počitka, da bi lahko svoje delo najbolje opravljali.

15. Spodbudite se k dejanjem

    bodise sam svoj navijač! V vsaki situaciji iščite dobro. Mislite na rešitev, ne na težave! Bodite optimistični in iznajdljivi.

16. Urite se v ustvarjalnem odlašanju

    ker vsega ne morete opraviti, se morate naučiti, da namenoma odlašate z nalogami, ki niso zelo pomembne. Tako boste imeli dovolj časa za tiste res pomembne.

17. Najprej opravite najtežjo nalogo

    Vsak delavnik začnite s svojo najtežjo in najpomembnejšo nalogo. Vztrajajte, dokler je popolnoma ne opravite.

18. Nalogo razdelite na manjše dele

    velike in zapletene naloge razdelite na manjše dele, potem začnite s prvim majhnim delom.

19. Vzemite si dovolj časa

    svoj delovni dan organizirajte tako, da boste imeli v njem obsežne bloke časa, v katerih se boste lahko popolnoma zbrali

20. Razvijte občutek nujnosti

    navaditese hitro opravljati svoje ključne naloge. Postanite znani kot človek, ki stvari opravi hitro in dobro.

21. Opravite eno nalogo naenkrat

    postavite si jasne prioritete in se takoj lotite svoje najpomembnejše naloge. Brez prestanka delajte in vztrajajte, dokler naloga ni v celoti izpolnjena.To je pravi ključ do izjemne kakovosti dela in popolne osebne storilnosti.

Poglavje 29

Obvladovanje osebne produktivnosti po Davidu Allenu

29.1 Stres zaradi preobilja obveznosti

- preveč stvari za narediti, premalo časa
- izboljšala se je kvaliteta življenja, toda preveč obveznosti,
- narava dela se je spremenila, več intelektualnega dela
- včasih je bilo delo bolj vidno, evidentno
- naše delo nima jasne meje
- prepletanje zasebnega in poklicnega dela
- organizacije se stalno spreminjajo
- ljudje bolj pogosto spreminjajo delo, kariero, se morajo izobraževati
- preobremenjenost z informacijami
- naša izobrazba je glede tega pomanjkljiva
- tradicionalne metode in orodja za upravljanje s časom so neustrezna

29.2 Kaj je zaželjeno stanje

- koncentracija, fokusiranje,
- če ne reagiramo ali če pretirano reagiramo na kaj, nas to po nepotrebnem kontrolira
- kako se spraviti v produktivno mentalno stanje?
- glavni vzrok za stres: neprimerno upravljanje z obveznostmi, ki jih sprejmemo
- vsaka nedokončana obveznost nas obremenjuje zavedno ali nezavedno — odprte zanke
- lahko so to velike ali male obveznosti, službene ali privatne

29.3 Princip: učinkovito ravnanje z notranjimi zadolžitvami

1. če je v glavi, v glavi ni jasno, zato vse kar moramo narediti, moramo zajeti v nek zanesljiv zunanji sistem
2. za vsako obveznost moramo jasno ugotoviti, kaj moramo narediti (če sploh kaj), da bi jo lahko izpolnili
3. ko ugotovimo, katere akcije so potrebne, si jih moramo zabeležiti in jih redno pregledovati.

29.4 Eksperiment:

- opisi projekt ali situacijo, ki te trenutno najbolj žuli!
- zapisi prvo fizično akcijo, ki je potrebna, da se bo projekt ali situacija odvila naprej!
- Kaj se je spremenilo: fizično nič, vendar nam je bolj jasno, kaj hočemo in kaj moramo narediti najprej, da bi se to zgodilo
- To je rezultat razmišljanja!
- V današnjem svetu delo ni kar dano, z razmišsljanjem ga moramo določiti!

29.4.1 Zakaj nimamo čiste glave?

1. nismo do konca premislili, kaj naj bi bil konkreten cilj

2. nismo se odločili, kaj je naslednji fizični korak akcije

3. nismo zabeležili v zanesljiv sistem niti cilja, niti naslednje akcije

Nasa zavest nima še ene svoje zavesti. Da moramo kupiti nove baterije, se spomnimo, šele ko vzamemo v roke ročno svetilko, namesto da bi se spomnili prvič, ko gremo v trgovino. Kolikokrat vsak dan po nepotrebnem ponovno in ponovno razmišljamo, kaj vse moramo postoriti? Vse kar pride v našo zavest in se v njej zadržuje, bi morali transformirati v cilje in akcije!

29.5 Kaj lahko upravljamo: naše AKTIVNOSTI

Kot športniki se lahko streniramo tudi v intelektualnem delu. Ne upravljamo ne časa, ne informacijske preobremenjenosti, ne prioritet, upravljamo lahko le naše akcije:

- kaj narediti z nasim časom,

- kaj narediti z informacijami,

- kar naj naše telo in naša koncentracija naredi glede na naše prioritete

in za vse to imamo naše omejene zmogljivosti!

Pravo vprašanje upravljanje z nami je: kako pravilno izbirati, kaj narediti v vsakem trenutku?

Velik del občutkov nemoči, da izpeljemo stvari in da imamo premalo časa izvira iz tega, da nismo jasno ugotovili, kaj je naslednja potrebna akcija, da tečejo stvari naprej. Teoretično bi bil najbolj pravilen pristop k analizi, kaj je potrebno narediti, od zgoraj navzdol! Praktično pa je od spodaj navzgor bolj učinkovit.

Horizontalno in vertikalno upravljanje: horizontalno usklajuje vse aktivnosti, v katere smo vključeni. Vertikalno upravljanje pa v sklopu posameznih projektov, kaj je potrebno narediti v sklopu posameznega projekta.

Kratkoročni spomin je kot RAM z omejeno kapaciteto. Če ga želimo učinkovito izrabiti, mora biti čimbolj prazen.

29.6 Pet stopenj upravljanja z našimi aktivnostmi

1. Zbiranje stvari, ki zbudijo našo pozornost
2. Procesiranje kaj te stvari pomenijo in kaj naj z njimi naredimo
3. Organiziranje rezultatov, ki jih
4. ocenjujemo, da bi izbrali, katero naj
5. naredimo.

Ne poskušajmo delati vseh petih stvari naenkrat!

29.6.1 Zbiranje

- fizični nabiralniki, e-mail, SMS, telefon
- vse iz glave v nabiralnik!
- imejmo čimmanj nabiralnikov (problem je bolj izrazit pri papirčkih in zvezkih)
- nabiralnike moramo redno prazniti!

29.6.2 Procesiranje

Da se dejansko lahko organiziramo, ni dovolj da le zbiramo, ampak se moramo odločiti, kaj narediti s tem, kar smo zbrali.

1. Kaj je?
2. Ali sledi mozna akcija? DA / NE
3. Če NE:
   (a) v smeti
   (b) morda enkrat kasneje (inkubacija)
   (c) stvar je morda zanimiva kdaj kasneje - arhiviramo
4. Če DA:
   (a) ali ima zvezo z nekim projektom? (potem je potrebno to akcijo postaviti v kontekst projekta)

(b) katera je naslednja potrebna fizična akcija?

i. naredi jo takoj, če zahteva manj kot 2 minuti

ii. delegiraj jo

iii. odloži jo in jo daj na spisek naslednjih akcij

29.6.3 Organiziranje

- kot rezultat procesiranja imamo 8 diskretnih kategorij stvari
- to so lahko spiski na papirju, v koledarčku, na raznih napravah itd.
- Projekt je vsako doseganje ciljev, ki zahteva več kot eno akcijo.
- Naslednje akcije: koledarsko določene (akcija ob določenem času, akcija na določen dan, informacija na določen dan)

29.6.4 Ocenjevanje

- Vsebino naših kategorij moramo redno pregledovati in ocenjevati
- Koledar, spisek naslednjih akcij (lahko so razdeljene na področja: služba, doma, ob računalniku, v avtu . . .)
- Kritično za uspeh metode: tedenski pregled vseh kategorij!
- mnogo se nas počuti dobro preden gremo na počitnice tudi zato, ker pred tem pospravimo mizo (in glavo).
- Zakaj ne bi tega narediti redno vsak teden in se prav tako počutili olajšane?

29.6.5 Delo

Trije modeli za izbiranje akcije (dela):

1. model:
   štirje kriteriji: kontekst, razpoložljiv čas, razpoložljiva energija, prioriteta
2. model: delamo preddefinirano delo, delamo kar pride oziroma se zgodi, definiramo kaj moramo narediti
3. model: 6-nivojski model za določanje naših prioritet:

(a) naše življenje
(b) 3-5 letna vizija
(c) 1-2 letna vizija
(d) naše zadolžitve
(e) tekoči projekti
(f) tekoče akcije

29.7 Pet faz projektnega načrtovanja

- horizontalni fokus pri razmišljanju o delu in naslednjih akcijah zadošča v večini situacij
- vertikalni fokus, pa pomeni vpeljati elemente planiranja
- najbolj učinkovito je dokaj preprosto in neformalno naravno planiranje v naslednjih 5 korakih:

  1. definiraj namen in glavne omejitve — zakaj?
  2. vizualiziraj si dosežene cilje
  3. brainstorming
  4. organiziranje
  5. identificiranje naslednje akcije

Formalni pristop je velikokrat nenaraven zato ker:

1. ocenjujemo rešitve, ne da bi se zares vprašali za namenom projekta
2. skušamo strukturirati projekt ne da bi vedeli točno kaj želimo narediti
3. zato se velikokrat izogibamo planiranju.