Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osebni proces razvoja programske opreme (Personal Software Process - PSP)

Študijsko gradivo za interno uporabo pri predmetu Razvoj programskih sistemov 2

Ljubljana, 2007/08

Kazalo

Kazalo (nadaljevanje)

Osebni proces razvoja PO – PSP (ang. Personal Software Process)

Osebni proces razvoja (PSP) definira nadziran proces razvoja PO za inženirje v organizaciji za razvoj PO.

Konkretneje: množica metod, predlog in navodil o tem, kako načrtovati, meriti in voditi potek dela inženirja, ki dela v razvoju PO.

Implementira CMM principe na ravni osebnega dela inženirjev:

- podmožico primernih metod in praks so prilagodili ter sestavili v model
- programer postopoma razvija svoj proces:
  - začne z osnovnimi postopki
  - te postopoma nadgrajuje z uvedbo merjenja, principov za zagotavljanje kakovosti dela in aktivnosti za obvladovanje večje količine dela
  - slednjič doseže optimalen proces, ki mu omogoča popolno obvladovanje svojega dela

Osebni proces razvoja PO – PSP

Splošni cilj: pravočasen in poceni razvoj PO brez napak.

PSP principi:

- Vsak inženir je različen; da bi bil pri svojem delu kar najbolj učinkovit, mora skrbno načrtovati svoje delo na osnovi preteklih izkušenj.
- Izboljšanje učinkovitosti dela zahteva discipliniran proces razvoja PO z merjenjem izdelkov (velikosti, napak) in vloženega napora.
- Inženirji lahko izdelajo kakovostne izdelke samo tako, da se čutijo osebno zavezane in odgovorne za najprimernejši način dela, ki do tega vodi.
- Zgodnje odpravljanje napak v procesu je cenejše od kasnejšega; še ceneje je preprečiti nastanek napak.
- Najboljši način dela je tisti, ki najhitreje in najceneje vodi do rezultata.

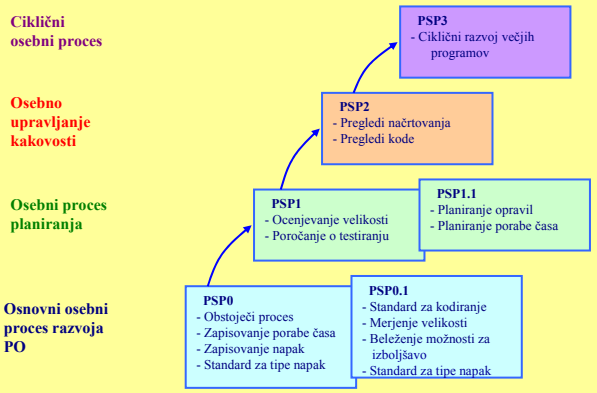
PSP ni omejen z uporabo programskega jezika ali metodologije dela.

Osebni proces razvoja PO – PSP

Sedem verzij modela PSP:

- osebni procesi od najbolj enostavnega (PSP0) do najbolj popolnega (PSP3)
- vsaka verzija obsega več področij
- vsaka višja verzija vsebuje tudi vsa področja nižje verzije
- inženir napreduje postopoma:
  - na začetku izbere proces PSP0
  - ko ga obvlada, bo njegov cilj PSP0.1,
  - ko obvlada tega napreduje proti PSP1 ...
- po analogiji CMM lahko verzije PSP razvrstimo v **4 nivoje zrelosti**:
  - **PSP0: Osnovni osebni proces razvoja PO**
  - **PSP1: Osebni proces planiranja**
  - **PSP2: osebno upravljanje kakovosti**
  - **PSP3: ciklični osebni proces**

Osebni proces razvoja PO – PSP



Osebni proces razvoja PO – PSP

1. nivo: PSP0: Osnovni osebni proces razvoja PO

- uvaja osnovni produkcijski osebni proces razvoja PO
- obsega verziji PSP0 in PSP0.1
- **PSP0:**
- vzpostavlja nadziran osebni proces
- vsebuje osnovna merjenja
- beleži osnovne rezultate, ki je osnova za kakršenkoli napredek
- Sestava procesa: planiranje, sledi produkcijski cikel in analizna faza
- produkcijski cikel: (obstoječi) življenski cikel razvoja PO z merjenjem
- analizna faza: analiza merskih rezultatov
- **PSP0.1:**
- dodatno uvajanja standard za kodiranje,
- pomembno: začnemo z merjenjem velikosti na ustrezen način
- uporablja tudi formular za beleženje možnosti za izboljšavo.

Osebni proces razvoja PO – PSP

2. nivo: PSP1: Osebni proces planiranja

- uvaja planiranje
- obsega verziji PSP1 in PSP1.1.
- **PSP1:**
- definira planiranje velikosti
- definira planiranje porabe virov,
- vsebuje vsa osnovna merjenja
- uvaja poročilo o testiranju
- **PSP1.1:**
- dodatno uvajanja planiranje opravil
- uvaja tudi planiranje porabe časa
- pomembno: metode za napovedovanje izidov (PROBE)

Osebni proces razvoja PO – PSP

3. nivo: PSP2: Osebno upravljanje kakovosti

- uvaja upravljanje kakovosti izdelkov in procesa
- obsega verziji PSP2 in PSP2.1.
- **PSP2:**
- uvaja tehnike za pregledovanje načrta
- uvaja tehnike za pregledovanje kode
- omogoča učinkovito odkrivanje napak,
- omogoča ocenjevanje kakovosti
- **PSP2.1:**
- dodatno uvajanja tehnike za preverjanje načrtov

Osebni proces razvoja PO – PSP

4. nivo: PSP3: Ciklični osebni proces

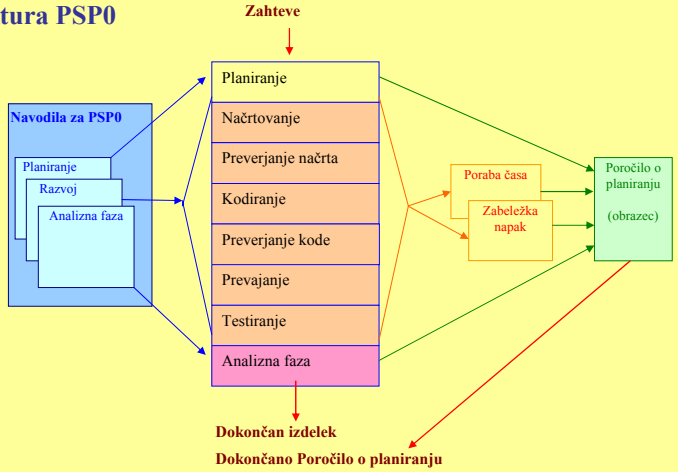
- dodaja tehnike za obvladovanje razvoja večjih programov
- obsega samo verzijo PSP3
- **PSP3:**
- cikličen razvoj programov – delitev na obvladljive dele

Oprerativno je vsaka verzija PSP-ja definirana z množico dokumentov, ki pripadajo enemu od štirih tipov:

- navodilo
- obrazec
- zapisek
- standard

Osebni proces razvoja PO – PSP

Struktura PSP0



Osebni proces razvoja PO – PSP

Različne verzije PSP-ja je definirana z različnim številom dokumentov:

- PSP0: 11 dokumentov (navodil, obrazcev, zapiskov in standardov)
- PSP1: 19 dokumentov
- PSP2: 25 dokumentov
- PSP3: 39 dokumentov.

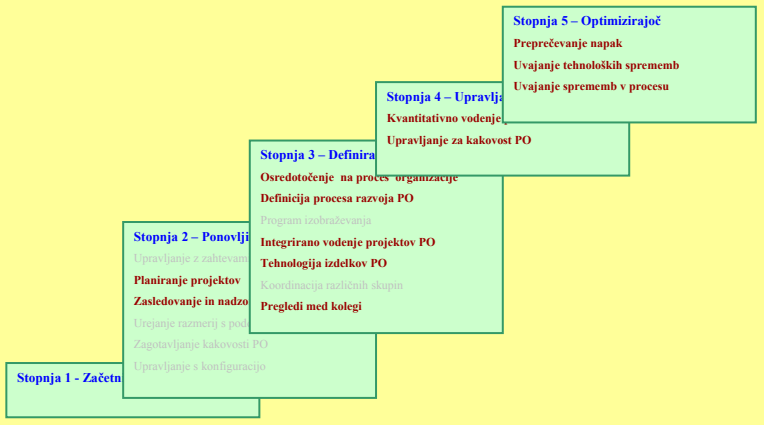
PSP lahko z vidika posameznika nadomesti CMM:

- pokrita so vsa ključna področja nivojev 4 in 5 (kakovost!),
- manjkajo posamezna področja iz 2. in 3. nivoja (projekti, organizacija)

Velja:

- PSP je lahko namenjen projektom, pri katerih sodeluje en sam razvijalec
- če je razvijalcev več, potrebujemo vsaj TSP

Osebni proces razvoja PO – PSP



PSP: Upravljanje s časom

Način upravljanja s časom:

- **Osnovno načelo:** zelo verjetno je, da bomo ta teden porabili približno enako časa za ista opravila kot smo jih prejšnji teden
- da bi lahko realistično planirali porabo časa, moramo spremljati dejansko porabo časa
- da bi lahko spremljali točnost svojih ocen porabe časa in časovnih planov, moramo te dokumentirati in pozneje primerjati z dejansko porabo časa
- plane je treba postopoma izboljševati: odkrivati predhodne napake in možnosti za izboljšave

Bistvo upravljanja s časom: določimo plan porabe časa in se ga nato skušamo čimbolj držati

Pridobitve: odkrijemo problematično porabo časa, postopoma izboljšamo sposobnost napovedovanja, spremenimo svoj običajen način dela

PSP: Upravljanje s časom

Kako porabimo svoj čas:

a) določimo glavne skupine aktivnosti

- upoštevamo vidik, ki nas zanima
- 3 do 5 kategorij
- če potrebujemo večjo natančnost, določimo podkategorije

b) beležimo porabljen čas po glavnih aktivnostih

- beležimo začetek in konec izvajanja aktivnosti
- natančnost – v minutah
- beleži dejanski čas (izgube)

c) beležimo čas na standardiziran način

- kratko in jedrnato
- tabela – zabeležka

d) hranimo podatke o porabljenem času na prikladnem mestu

PSP: Upravljanje s časom

“Inženirjev zvezek”:

a) **Namen:** pripomoček in beležka za zapis:

- delovnih nalog in opravil
- podatkov o porabi časa za posamezna opravila
- rešitev ali idej,
- načrtov, planov ipd.

b) Lastnik je posameznik (lahko skupina)

c) Omogoča spremljanje (ali nadzor) dela inženirja, tudi časovni

e) **Sestava:**

- prva stran: številka, naslov, ime, priimek, oddelek, telefon, datum začetka in konca zapisov)
- kazalo vsebine: logična struktura, strani
- poglavja: splošna in vsebinsko ločena
- zapisi: kratki in jednati, pregledni (barve, zamiki)

PSP: Spremljanje porabe časa

- **Izkušnja:** zakaj spremljati čas?
- **Predlagan način: zakaj bi ga uporabljali?**
  - standard – vsi enako spremljamo čas
  - brez izkušenj si je težko zamisliti primeren način
  - po koncu predavanj ga boste prilagodili vašim potrebam
- **Merjenje časa mora biti natančno:**
  - izkušnje kažejo, da so ure pregroba enota
  - merimo do minute natančno
  - prikladno: računalniška ura, lahko drugi pripomočki
  - nepolne ure: decimalke ali minute?
  - **rešitev: merimo vse samo v minutah!**

Tipično: spremljanje dnevnih aktivnosti: spanje, hrana, delo, potovanje, telovadba, zabava ...

PSP: Spremljanje porabe časa

Kaj dejansko merimo/ beležimo:

- **Datum**: datum izvajanja aktivnosti (dd.mm.[llll])
- **Začetek**: začetek izvajanja aktivnosti (uu:mm)
- **Konec**: konec izvajanja aktivnosti (uu:mm)
- **Prekinitev**: čas prekinitev med delom ([mm[+mm[+...]]])
- **Dej.čas**: dejanski čas izvajanja aktivnosti (mm)
- **Aktivnost**: ime kategorije aktivnosti ali kratek naziv aktivnosti
- **Opis**: podrobnejši opis, komentar
- **Z**: ali je aktivnost zaključena (x/)
- **E**: število enot opravljenega dela, če je aktivnost zaključena

PSP: Spremljanje porabe časa

Zaključek izvedbe aktivosti:

- lahko beležimo ali ne, če aktivnost nima posebnega rezultata in je čas izvajanja aktivnosti ustrezno merilo obsega aktivnosti
- če beležimo lahko okvirno pretvorimo na večje enote (šolske ure)
- zaključek je treba beležiti, ko je rezultat merljiv
- vpliv velikosti enote (branje: stran ali poglavje)

Obseg opravljenega dela:

- vnašamo samo takrat, ko je aktivnost zaključena
- različne (kategorije) aktivnosti imajo različne enote
- tipično: čas*, velikost rezultata (število vrstic prog.kode, število strani, število elementov diagrama ipd.), pridobljena vrednost, pridobljen delež do končnega cilja

PSP: Spremljanje porabe časa

Primer kategorij aktivnosti pri študiju predmeta RPS2:

- **predavanja:** obisk predavanj (enota: 1 šolska ura = 45 minut)
- **av.vaje:** obisk avditornih vaj (enota: 1 šolska ura = 45 minut)
- **lab. vaje:** obisk laboratorijskih vaj (enota: 1 šolska ura = 45 minut)
- **branje:** branje štud. literature, ponavljanje snovi po zapiskih (enota: 1 stran knjige, zvezka)
- **račun:** reševanje (računskih) nalog, načrtovanje – diagramske tehnike (enota: 1 šolska ura = 45 minut)
- **program:** programiranje, razvoj delov aplikacije – recimo v portalu (enota: 1 vrstica programske kode*)
- **internet:** študij s pomočjo interneta (enota: 1 šolska ura = 45 minut)

PSP: Spremljanje porabe časa

Primer zapisa (beležke) porabe časa:

Andrej Novak **Datum začetka: 07.03.2005**

Namen: Študij predmeta RPS2 **Zap.št.zapisa: 1. teden**

Datum	Začetek	Konec	Prekinitev	Čas	Aktivnost	Opis	Z	E
07.03.	10:15	12:00		105	Lab.vaje	Oracle potral 2: Uvod	x	2
	19:10	20:02		52	Branje	Fenton: SW Metrics (1.pog)	x	22
	20:02	21:30	14	74	Program	Prijava v testni portal, uporaba		
09.03.	8:17	8:33		16	Branje	Ponovitev snovi pret.tedna (zvezek)		
	11:15	12:00		45	Av.vaje	Avditorne vaje – uvod v portal	x	1
	19:31	20:42	7	64	Branje	Ponovitev snovi pret.tedna (zvezek)	x	13
10.03.	7:30	10:00	15	135	Predavanje	Merjenje v TPO	x	3
	18:36	20:52	12+30+7	87	Branje	Fenton: SW Metrics (2.pog)	x	23
12.03.	10:10	11:53	15*	88	Internet	SEI: CMM	x	2

PSP: Spremljanje porabe časa

Nasveti:

- porabo časa beleži v inženirski zvezek
- inženirski zvezek imej vedno pri sebi
- skušaj vedno sproti beležiti porabljen čas
- če pozabiš zabeležiti porabo, vnesi čim boljšo oceno porabljenega časa (z zvezdico) takoj ko se spomneš
- če imaš težave z merjenjem časa, uporabi štoparico
- redno seštevaj porabljen čas po opravilih (opravljena naloga, zaključen tečaj) in zaključenih časovnih enotah (tednih)
- uporaba računalnika je manj praktična!

PSP: Časovni plan in plan doseganja cilja

Spremljanje porabe časa nam omogoča planiranje:
Dve vrsti planov:

- **Časovni plan:**
- definiran za določeno časovno obdobje (dan, teden, mesec, leto)
- planiramo porabo časa v časovnem obdobju z razporejanjem nabora predvidenih aktivnosti
- časovno obdobje razdelimo na manjše enote
- nabor aktivnosti je znan
- vsaki enoti pripišemo primerno aktivnost

Pomembno:

- celovit pogled na to, kaj moramo v celotnem obdobju doseči
- optimalno razporedimo aktivnosti, da čimbolje izkoristimo čas

PSP: Časovni plan in plan doseganja cilja

b) Plan doseganja cilja

- določajo časovni razpored aktivnosti, ki vodi do doseganja cilja
- cilj je lahko:
  - **konkreten:** udeležiti se predavanja 17.3.ob 7.30, prebrati knjigo N.Fentona: SW Metrics, izdelati 30 strani dolgo poročilo ...
  - **splošen:** naučiti se uporabljati Oraclov Potral 2, izvajati kakovostne lab.vaje pri predmetu RPS2 ...
- plan določa, kako izvesti ustrezno zaporedje aktivnosti, ki vodi do rezultata

Pomembno:

- bistveno je, **kako** bomo dosegli cilj
- s tem opredelimo **kdaj** bo cilj dosežen (**kolikšni** bodo stroški ...)
- šele na osnovi teh informacij lahko določimo časovni plan

PSP: Časovni plan in plan doseganja cilja

Primer povezave med časovnim planom in planom doseganja cilja:

- prebrati želimo knjigo (200 strani, 20 poglavij)
- ocena: za eno poglavje potrebujemo povprečno 1 uro

Plan doseganja cilja:

- cilj je knjigo zares prebrati v 20 urah (oziroma zaključiti branje v 10 tednih)
- predvidimo, da bomo knjigo brali 2 uri na teden

Časovni plan:

- cilj je optimalno razporediti dve uri branja na teden med ostale tedenske aktivnosti
- **vendar:** ravnamo se po časovnem planu: ta realistično (celovito) določa vse aktivnosti, upošteva naša pravila, določa razmerje med delom in počitkom: tudi to, da lahko beremo knjigo 2 uri na teden

PSP: Časovni plan in plan doseganja cilja

PSP: Časovni plan in plan doseganja cilja

Tedenski pregled aktivnosti: tabele s seštevki časa po dnevih/tednih, ki nastanjo na podlagi spremljanja porabe časa skozi daljše obdobje

- primer časovnega plana
- obrazec !
- (nujna) nadgradnja spremljanja časa
- tipično: vse dnevne aktivnosti (spanje, hrana, delo, telovadba, zabava)
- lahko določeno področje aktivnosti (študij predmeta RPS2)
- določimo jih ob zaključku tedna (vedno na isti dan v tednu)

Celovit pregled porabe časa z vidika:

- posameznih kategorij aktivnosti
- posameznih dni v tednu
- vsote porabe časa z vidika kategorij aktivnosti
- povprečnih vrednosti, minimumov in maksimumov
- celotne vsote porabljenega časa

PSP: Časovni plan in plan doseganja cilja

Tri tabele:

a) Tabela vsote časov, ki smo ga **posamezne dni** izbranega tedna namenili za posamezne kategorije aktivnosti

b) Tabela vsote časov, ki smo ga v predhodnem tednu **skupaj** namenili za posamezne kategorije aktivnosti

c) Tabela vsote časov, ki smo ga v tekočem tednu **skupaj** namenili za posamezne kategorije aktivnosti

Ogledali si bomo primere tabel po 1. tednu!

PSP: Časovni plan in plan doseganja cilja

a) Tabela vsote časov, ki smo ga posamezne dni izbranega tedna namenili za posamezne kategorije aktivnosti

- **Kategorije aktivnosti:** predavanja, avditorne vaje, laboratorijske vaje, branje (literature), račun(anje oz reševanje nalog), program(iranje), (študij s pomočjo) internet(a)
- **Dnevi v tednu** z datumi (npr. sreda, 17.6.)
- **Vsota časa v določeni kategoriji na določen dan** (v minutah)
- **Vsota vsega časa v tednu po dnevih** (v minutah)
- **Vsota vsega časa v tednu po kategorijah** (v minutah)
- **Vsota vsega porabljenega časa v tednu** (v minutah)

PSP: Časovni plan in plan doseganja cilja

Primer beležke porabe časa:

Andrej Novak — Datum začetka: 07.03.2005

Namen: Študij predmeta RPS2 — Zap.št.zapisa: 1. teden

Datum	Začetek	Konec	Prekinitev	Čas	Aktivnost	Opis	Z	E
07.03.	10:15	12:00		105	Lab.vaje	Oracle potral 2: Uvod	x	2
	19:10	20:02		52	Branje	Fenton: SW Metrics (1.pog)	x	22
	20:02	21:30	14	74	Program	Prijava v testni portal, uporaba		
09.03.	8:17	8:33		16	Branje	Ponovitev snovi pret.tedna (zvezek)		
	11:15	12:00		45	Av.vaje	Avditorne vaje – uvod v portal	x	1
	19:31	20:42	7	64	Branje	Ponovitev snovi pret.tedna (zvezek)	x	13
10.03.	7:30	10:00	15	135	Predavanje	Merjenje v TPO	x	3
	18:36	20:52	12+30+7	87	Branje	Fenton: SW Metrics (2.pog)	x	23
12.03.	10:10	11:53	15*	88	Internet	SEI: CMM	x	2

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak — Datum začetka: 07.03.2005

Namen: Študij predmeta RPS2 — Zap.št.tedna: 1. teden

Dan\Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
PON,7.3.			105	52		74		231
TOR,8.3.								
SRE,9.3.		45		80				125
ČET,10.3.	135			87				222
PET,11.3.								
SOB,12.3.							88	88
NED,13.3.								
SKUPAJ	135	45	105	219		74	88	666

PSP: Časovni plan in plan doseganja cilja

b) Tabela vsote časov, ki smo ga v predhodnem tednu skupaj namenili za posamezne kategorije aktivnosti

- **Skupaj**: Skupna vsota porabljenega časa do konca predhodnega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Povp(rečno)**: povprečna vsota porabljenega časa na teden, ki je veljala ob koncu predhodnega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Min(imalno)**: najmanjša vsota porabljenega časa na teden, ki je veljala ob koncu predhodnega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Max(imalno)**: največja vsota porabljenega časa na teden, ki je veljala ob koncu predhodnega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Skupno število tednov**, ki je upoštevano v tabeli

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 07.03.2005
Zap.št.tedna: 1. teden

SEŠTEVEK ČASOV PREJŠNJEGA TEDNA

Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
SKUPAJ								
POVPR.								
MAX.								
MIN.								

Skupno število tednov:

PSP: Časovni plan in plan doseganja cilja

c) Tabela vsote časov, ki smo ga v tekočem tednu skupaj namenili za posamezne kategorije aktivnosti

- **Skupaj**: Skupna vsota porabljenega časa do konca tekočega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Povp(rečno)**: povprečna vsota porabljenega časa na teden, ki je veljala ob koncu tekočega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Min(imalno)**: najmanjša vsota porabljenega časa na teden, ki je veljala ob koncu tekočega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Max(imalno)**: največja vsota porabljenega časa na teden, ki je veljala ob koncu tekočega tedna po posameznih kategorijah aktivnosti in skupaj (v minutah)
- **Skupno število tednov**, ki je upoštevano v tabeli

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 07.03.2005
Zap.št.tedna: 1. teden

SEŠTEVEK ČASOV TEKOČEGA TEDNA

Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
SKUPAJ	135	45	105	219		74	88	666
POVPR.	135	45	105	219		74	88	666
MAX.	135	45	105	219		74	88	666
MIN.	135	45	105	219		74	88	666

Skupno število tednov: 1

PSP: Časovni plan in plan doseganja cilja

Primer: tabele po drugem tednu

- naslednji teden prištejemo rezultate
- pomembno: izračun posameznih elementov tabele vsot časov
- uporaba tabele predhodnih časov – praktično za izračun

Pravi učinek se pokaže po daljšem času!

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 14.03.2005
Zap.št.tedna: 2. teden

Dan\Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
PON,14.3.			105	11				116
TOR,15.3.							74	74
SRE,16.3.		45		40				85
ČET,17.3.	135							135
PET,18.3.								
SOB,19.3.					42			42
NED,20.3.								
SKUPAJ	135	45	105	51	42		74	452

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 14.03.2005
Zap.št.tedna: 2. teden

SEŠTEVEK ČASOV PREJŠNJEGA TEDNA

Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
SKUPAJ	135	45	105	219		74	88	666
POVPR.	135	45	105	219		74	88	666
MAX.	135	45	105	219		74	88	666
MIN.	135	45	105	219		74	88	666

Skupno število tednov: 1

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 14.03.2005
Zap.št. tedna: 2. teden

SEŠTEVEK ČASOV TEKOČEGA TEDNA

Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
SKUPAJ	270	90	210	270	42	74	162	1118
POVPR.	135	45	105	135	21	37	81	559
MAX.	135	45	105	219	42	74	88	666
MIN.	135	45	105	51			74	452

Skupno število tednov: 2

PSP: Časovni plan in plan doseganja cilja

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 06.06.2005
Zap.št. tedna: 13. teden

SEŠTEVEK ČASOV TEKOČEGA TEDNA

Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
SKUPAJ	1755	585	1365	1270	280	810	353	6418
POVPR.	135	45	105	98	22	62	27	494
MAX.	135	45	105	269	203	375	128	861
MIN.	135	45	105					312

Skupno število tednov: 13

PSP: Časovni plan in plan doseganja cilja

Praktični nasveti:

- vodenje evidence porabe časa je zamudno in naporno!
- ne pozabi na smisel spremljanja časa!
- od začetka moramo nekaj časa dosledno upoštevati obrazec
- čez nekaj časa razumemo zakonitost porabe svojega časa
- tedaj lahko spremenimo pristop in začnemo spremljati čas drugače - recimo samo določene aktivnosti (recimo branje in računanje), uporabimo daljši interval (mesec)
- osredotočimo se na cilj, ki ga želimo doseči
- računalniška orodja za delo s tabelami pomagajo (in tudi ne)
- tudi po daljšem obdobju uporabe obrazca, ko dodobra poznamo zakonitosti svoje porabe časa, je še vedno (okvirno) smiselno spremljati porabo časa
- moja izkušnja

PSP: Planiranje izdelkov

Časovne plane uporabimo kot podlago za planiranje izdelkov

Plan izdelka = plan doseganja cilja (izdelati izdelek)

Zakaj planirati?

- IBM primer...
- planiranje izdelkov ni nič novega: podjetja to počnejo od nekdaj
- če ne planiramo izdelkov:
  - ne poznamo (razumemo) svojega statusa
  - ne znamo predvideti, če bomo dosegli cilj
  - ne poznamo možnosti, kaj vse lahko dosežemo
- čeprav delamo v razvojni skupini potrebujemo **svoje** plane
- planirajmo vse pomembnejše izdelke (cilje)
- pomembno je določiti **smiselne** in **natančne** plane

PSP: Planiranje izdelkov

Prvi korak: jasna definicija izdelka (rezultata)

Trije ključni deli:

- velikost (in ostale pomembne značilnosti) izdelka
- ocena časa, ki bo potreben za izdelavo izdelka
- približen časovni plan izdelave

Dodatno za zahtevnejše primere:

- odgovornost,
- izvajalci,
- podrobne specifikacije,
- odvisnosti od ostalih izdelkov,
- posebne zahteve za kakovost

Tipični primeri: program, načrt PB, plan za testiranje

PSP: Planiranje izdelkov

Pristop:

a) Planiranja se učimo postopoma:

- zastavimo plan,
- preverimo rezultate,
- izboljšamo pristop

b) Začnemo s planiranjem malih aktivnosti

- majhne aktivnosti = enostavni plani
- majhne aktivnosti – pogosta izvedba - znamo oceniti čas za izvedbo

c) Več majhnih aktivnosti sestavlja veliko aktivnost

- več majhnih planov sestavimo v kompleksen plan
- pomembna je sestava večje aktivnosti

PSP: Planiranje izdelkov

a) Napiši program, ki:

- izpiše, ali je podano število praštevilo
- izpiše vsa praštevila med spodnjo in zgornjo mejo
- omogoča izvedbo različnih matematičnih izračunov

b) Preberi:

- eno stran v knjigi
- poglavje v knjigi, ki je v povprečju dolgo 10 strani
- knjigo (150 – 250 strani)

c) Napiši:

- en odstavek besedila (cca 5 vrstic)
- eno stran besedila (60 vrstic)
- poglavje (cca 8 strani)
- diplomsko nalogo (30 – 50 strani)

PSP: Planiranje izdelkov

Definicije:

- **Izdelek (product):** nekaj, kar izdelamo po naročilu nekoga (stranke, nadrejenega,sodelavca), otipliv rezultat našega dela
- **Projekt (project):** zaporedje aktivnosti in virov, ki jih izvajamo oziroma porabljamo pri izdelavi izdelka
- **Aktivnost (task, activity):** definiran element dela
- **Proces (process):** definira način, kako izdelati izdelke
- **Plan (plan):** opisuje način kako izvajati projekte: kdo, kako, kdaj in za kakšno ceno
- **Opravilo (job):** nekaj, kar počnemo: lahko projekt ali aktivnost

Odslej namesto termina **aktivnost** raje uporabljamo termin **opravilo**!

Razmerje med **kategorijo aktivnosti** in **vrsto opravil**

PSP: Planiranje izdelkov

“Beležka opravil”

Beležimo

a) vrsto in opis opravil,
b) plan opravil,
c) dejansko opravljeno delo ter
d) razmerje med prvim in drugim

Tipičen primer plana izdelka, ki dopolnjuje plan porabe časa

Beležka, ne obrazec!

Prikaz učinkovite uporabe aktualnih podatkov pri planiranju: na podlagi izmerjenih vrednosti za isto vrsto opravila planiramo / ocenjujemo nadaljnja opravila

Izberemo nabor zanimivih vrst opravil

PSP: Planiranje izdelkov

Elementi beležke:

 # Opr: zaporedna številka opravila
Datum: datum začetka izvajanja opravila
Vrsta: vrsta opravila (recimo branje, program)

Ocena: ocenjene oziroma planirane vrednosti (pred dejansko izvedbo opravil)
- **Čas:** ocena porabe časa za izvedbo opravila (v minutah)
- **Enot:** ocena velikosti izdelka, ko nastane ob izvedbi opravila (v ustreznih enotah, odvisno od vrste opravila)

Izmerjeno: dejansko izmerjene vrednosti (po izvedbi opravila)
- **Čas:** izmerjena vrednost porabe časa za izvedbo opravila (v minutah)
- **Enot:** velikost izdelka, ko nastane ob izvedbi opravila (v ustreznih enotah)
- **Raz:** razmerje med dejanskim časom in velikostjo

PSP: Planiranje izdelkov

Elementi beležke (nadaljevanje):

Doslej: skupne vrednosti za doslej opravljena opravila, grupirane po posameznih vrstah opravil (izpolnimo po izvedbi posameznega opravila)
- **Čas:** skupna poraba časa za izvedbo vseh opravil te vrste doslej (v minutah)
- **Enot:** skupna velikost izdelkov te vrste doslej (v ustreznih enotah)
- **Raz:** razmerje med skupnim časom in velikostjo - dejansko povprečno razmerje glede na doslej izmerjene vrednosti
- **Max:** doslej najvišje izmerjeno razmerje med časom in velikostjo
- **Min:** doslej najmanjše izmerjeno razmerje med časom in velikostjo

Opis: opis opravila iz ustrezne vrste opravil, zraven lahko tudi enota za izraz razmerja med časom in velikostjo izdelka

PSP: Planiranje izdelkov

Ime: **Datum začetka:**

# Opr	Datum	Vrsta	Ocena		Izmerjeno			Doslej				
			Čas	Enot	Čas	Enot	Raz	Čas	Enot	Raz	Max	Min
Opis:												
Opis:												
Opis:												
Opis:												

PSP: Planiranje izdelkov

Vsako opravilo beležimo dvakrat!

Pred opravilom vnesemo:

- številko opravila
- datum začetka izvajanja opravila
- vrsta opravila
- (lahko) kratek opis
- oceno velikosti izdelka
- oceno porabe časa v minutah - glede na skupno razmerje med porabo časa in velikostjo za isto vrsto opravil doslej

PSP: Planiranje izdelkov

Po opravilu vnesemo:

- opis opravila (po potrebi)
- porabljen čas (v minutah)
- dejansko velikost nastalega izdelka
- razmerje med časom in velikostjo izdelka
- skupni porabljen čas za izbrano vrsto opravil (k prejšnjemu skupnemu času prištejemo čas zadnjega opravila)
- skupno velikost izdelkov (k prejšnji skupni velikosti izdelkov za izbrano vrsto opravil prištejemo velikost zadnjega izdelka)
- razmerje med skupnim časom in velikostjo

PSP: Planiranje izdelkov

Po opravilu vnesemo še:

- Največje razmerje med časom in velikostjo doslej (če je trenutno razmerje večje od doslej največjega razmerja za to vrsto, zapišemo tega, sicer prepišemo doslej največjo vrednost)
- Najmanjše razmerje med časom in velikostjo doslej (če je trenutno razmerje manjše od doslej najmanjšega razmerja za to vrsto, zapišemo tega, sicer prepišemo doslej najmanjšo vrednost)

PSP: Planiranje izdelkov

Primer beležke porabe časa:

Andrej Novak — Datum začetka: 07.03.2005

Namen: Študij predmeta RPS2 — Zap.št.zapisa: 1. teden

Datum	Začetek	Konec	Prekinitev	Čas	Aktivnost	Opis	Z	E
07.03.	10:15	12:00		105	Lab.vaje	Oracle potral 2: Uvod	x	2
	19:10	20:02		52	Branje #1	Fenton: SW Metrics (1.pog)	x	22
	20:02	21:30	14	74	Program #2	Prijava v testni portal, uporaba		
09.03.	8:17	8:33		16	Branje #3	Ponovitev snovi pret.tedna (zvezek)		
	11:15	12:00		45	Av.vaje	Avditorne vaje – uvod v portal	x	1
	19:31	20:42	7	64	Branje #3	Ponovitev snovi pret.tedna (zvezek)	x	13
10.03.	7:30	10:00	15	135	Predavanje	Merjenje v TPO	x	3
	18:36	20:52	12+20+7	97	Branje #4	Fenton: SW Metrics (2.pog)	x	23
12.03.	10:10	11:53	(15)	88	Internet	SEI: CMM	x	2

PSP: Planiranje izdelkov

Ime: Andrej Novak — Datum začetka: 07.03.2005

# Opr	Datum	Vrsta	Ocena		Izmerjeno			Doslej				
			Čas	Enot	Čas	Enot	Raz	Čas	Enot	Raz	Max	Min
1.	7.3.	branje	44	22	52	22	2.36	52	22	2.36	2.36	2.36
	Opis: Fenton: SW Metrics (1.pog) (minut na stran)											
2.	7.3.	prog.	100	100								
	Opis: Prijava v testni portal, uporaba (minut na ekvivalent vrstice kode)											
3.	9.3.	branje	31	13	80	13	6.15	132	35	3.77	6.15	2.36
	Opis: Ponovitev snovi pret.tedna (zvezek)											
4.	9.3.	branje	87	23	97	23	4.22	229	58	3.95	6.15	2.36
	Opis: Fenton: SW Metrics (2.pog)											

PSP: Planiranje izdelkov

Praktični nasveti:

- ob prvi oceni porabe časa ugibamo, velikost (včasih) lahko izmerimo
- kasneje uporabljamo zadnje skupno razmerje za opravilo iste vrste, vendar lahko to tudi spremenimo, če sodimo, da je to smiselno
- pri tem se poslužujemo največjega in najmanjšega razmerja, razmerja za najbolj podobna opravila v preteklosti in podobno
- pametno je zanimive aktivnosti označevati z zaporedno številko tudi v beležki porabe časa – boljši pregled in usklajenost
- izpolnjevanje beležke opravil zahteva računanje – priporočena uporaba računalniških orodij (spreadsheet)
- moje izkušnje ...

PSP: Planiranje izdelkov

Bistvo:

- **učinkovito planiranje**: planiramo majhna opravila,ta seštevamo v večja in dobimo večje plane
- **uravnoteženi plani**: ocena je previsoka približno tolikokrat kot je prenizka; napake se izničijo, plan je realen
- upoštevanje povprečnega razmerja je le ena, ne nujno najboljša metoda za doseganje uravnoteženih planov
- dolgotrajna, konsistentna uporaba istega načina planiranja nad istovrstnimi opravili vodi do optimalnih rezultatov
- **smiselno grupiranje vrst opravil**: včasih je treba istovrstno opravilo podrobneje deliti (branje zvezka, branje knjige v slovenščini, branje zahtevnega članka v angleščini)

PSP: Velikost izdelkov

Določanje velikosti je pomemben del proces planiranja:

- osnovni cilj je določiti porabo časa
- tega ocenjujemo na podlagi predhodnih vrednosti
- za pravilno upoštevanje razmerja potrebujemo oceno velikosti

Določanje velikosti je lahko zahtevno in nentančno (ne vedno)

Različni izdelki imajo različna merila velikosti

Izdelku lahko merimo velikost na različne načine

Knjiga - enota za velikost je:

- poglavje
- stran
- razdelek (odstavek, formula, tabela, slika)

Izberemo smiselno stopnjo podrobnosti !

PSP: Velikost izdelkov

Primer: Branje knjige po poglavjih

Poglavje	Čas branja	Število strani	Minut / stran
1.	40	9	40/9 = 4.44
2.	40	11	3.64
3.	28	12	2.33
4.	118	16	7.38
5.	71	17	4.18
6.	40	12	3.33
Skupaj	337	77	
Povprečje	56.17	12.83	4.38

PSP: Velikost izdelkov

Pri merjenju velikosti velja tudi:

- velikost mora upoštevati različne stopnje zahtevnosti opravila:
  branje članka ali branje stripa
- upoštevati je treba tudi jezik besedila
- znana opravila opravimo učinkoviteje: ponovno branje
- poraba časa je odvisna od namena: pregled ali podrobno branje članka
- poraba časa je odvisna tudi od drugih okoliščin: utrujenost, nujnost, razpoloženje, ipd.

Rešitev: posebej določamo velikost za posamezne (pod)vrste opravil

PSP: Velikost izdelkov

Merjenje velikosti programov:

- Število vrstic programske kode (LOC, KLOC)
- Nujno določiti standard - recimo:
  - ne štejemo praznih vrstic
  - ne štejemo komentarjev
  - ostalo (način pisanja zank, begin-end stavkov,...) uporabljamo vedno na enak način
- Vpliv programskega jezika: primerno je primerjati velikost programov v istem programskem jeziku
- Vpliv izkušenj: na začetku so časi daljši, pozneje krajši
- Primerno je poznati izmerjene vrednosti (in ocene) za zadnjih nekaj meritev

PSP: Velikost izdelkov

Primer: Razvoj šestih majhnih programov v Javi

Program	Čas razvoja	LOC	Minut / LOC
1.	158	20	158/20 = 7.90
2.	69	11	6.27
3.	114	14	8.14
4.	93	10	9.30
5.	95	14	6.79
6.	151	18	8.39
Skupaj	680	87	
Povprečje	110.0	14.5	7.82

PSP: Velikost izdelkov

Določeni programski izdelki niso neposredno merljivi:

- menuji
- datoteke (tabele)
- poročila
- zasloni

Drugače merimo izdelke, ki predstavljajo programe, a niso programska koda
Razlikovati je treba med napisano in generirano kodo

Rešitev:

- konsistentno uporabimo drugačna merila
- določimo pretvorbo v LOC

PSP: Velikost izdelkov

Ocenjevanje velikosti programov

- Težava: nimamo vnaprej znanega izdelka
- Ocenjujemo na najboljši način, kot znamo – veliko presoje
- Odločamo se na podlagi (vnaprej znane) značilnosti/vsebine programa

Glavna značilnost programa:	Zahtevnost (obsežnost):
FOR zanka	enostavna (majhna)
WHILE zanka	srednja (večja)
REPEAT zanka	zahtevna (velika)
IF pogoj	zelo zahtevna (zelo velika)
CASE pogoj	
Izračun	
Podatkovna struktura	

PSP: Velikost izdelkov

Primer: Razvoj osmih majhnih programov v Javi (po gl. značilnostih)

Program	Čas razvoja	LOC	Min / LOC	Zahtev.	Značilnost
2.	69	11	6.27	enost.	FOR
5.	95	13	7.30	sred.	FOR
8.	110	15	7.33	sred.	REPEAT
6.	151	18	8.39	zaht.	REPEAT
1.	138	20	6.90	sred.	IF
4.	93	11	8.45	zelo zaht.	CASE
7.	113	15	7.53	enost.	izračun
3.	114	14	8.14	večja	Pod.str.
...	...	...	...	...	...

PSP: Velikost izdelkov

Veliki programi:

Vsebujejo več elementov (zanke, pogoje, izračune ....)

Možen pristop:

- analiziramo zahteve in ugotovimo značilnosti, ki jih bo vseboval program
- pripravimo podatke o pretekli porabi časa, grupirani po značilnostih
- na tej osnovi ocenimo velikost programa

Lahko imamo več vrst podatkov za različne tipe programov

Z več vrstami rešujemo tudi ocene različnih programskih jezikov

Težava: historični podatki so za majhne programe - sinergičen učinek različnih značilnosti pri velikih programih

Delna rešitev: najmanjše, največje in povprečne vrednosti podatkov + presoja

Praksa: dobre ocene, če je program dobro analiziran

Z veliko meritvami (izkušnjami) pridobimo občutek!

PSP: Velikost izdelkov

Primer:

Program za izračun povprečne ocene študija za vse študente UNI RI študija FRI, ki so ta trenutek vpisani na smer Programska oprema

Denimo, da so podatki, ki jih potrebujemo za izračun, že pripravljeni:

- za vsakega študenta potrebujemo njegovo trenutno povprečno oceno
- ocene so zapisane na dve decimalke natančno
- imena niso pomembna – povprečne ocene so zgolj zaporedno zapisane
- podatki so grupirani po letnikih (2, 3, 4 in 5 letnik)

Primerna podatkovna struktura je dvodimenzionalna tabela

- prva dimenzija je 4 (število letnikov)
- druga dimenzija 200 (največje možno število študentov PO v letniku)

PSP: Velikost izdelkov

	2. letnik	3. letnik	4. letnik	5. letnik
1.	5.00	6.12	7.50	
2.	7.50	8.33	8.02	
3.	9.12	10.00	8.77	
4.	5.00	6.55	8.00	
37.	7.13	8.22	6.45	
38.	7.99		9.00	
84.	8.34		6.32	
85.			8.56	
200.				

PSP: Velikost izdelkov

Postopek: Obdelati moramo vse (pozitivne) povprečne ocene študentov

Dvojna zanka:

- zunanja FOR po letnikih
- notranja REPEAT po študentih letnika (mogoče ni nobenega študenta)

Pogoj:

- upoštevamo samo povprečne ocene študentov, ki so že opravili izpit

Izračun:

- (inicializacija)
- štetje pozitivnih povprečnih ocen
- prištevanje posameznih povprečnih ocen vsoti
- izračun povprečja

Izpis: zanemarimo ...

PSP: Velikost izdelkov

Primer: ocena velikosti za naš program

Vrsta	Min/LOC	Zahtev.	Značilnost	Min_{LOC}	Ave_{LOC}	Max_{LOC}
Zanka						
1x	6.27	enost.	FOR ali WHILE	7	8	16
1x	7.33	sred.	REPEAT	8	10	22
Pogoj						
1x	6.90	sred.	IF	7	10	13
Logika						
3x	7.53	enost.	izračun	5	7	12
OCENA				37	49	87

Komentar: program za izračun povprečne pozitivne ocene vseh študentov PO na UNI RI - dvodimenzionalna tabela (študent x letnik)

PSP: Velikost izdelkov

S pomočjo ocene velikosti lahko ocenimo tudi čas:

- **Min:** 7*6.27 + 8*7.33 + 7*6.90 + 3*5*7.53 = **263.78 minut**
- **Ave:** 8*6.27 + 10*7.33 + 10*6.90 + 3*7*7.53 = **350.59 minut**
- **Max:** 16*6.27 + 22*7.33 + 13*6.90 + 3*12*7.53 = **619.94 minut**

Oceno velikosti uporabimo v drugih obrazcih:

- **ob spremljanju porabe časa**
  - zaključene aktivnosti – enote velikosti izdelka
- **beležke opravil**
  - planirane velikosti izdelkov in poraba časa

PSP: Velikost izdelkov

Primer uporabe ocenjevanja velikosti v beležki opravil

Ime: Andrej Novak **Datum začetka: 07.03.2005**

# Opr	Datum	Vrsta	Ocena		Izmerjeno			Doslej				
			Čas	Enot	Čas	Enot	Raz	Čas	Enot	Raz	Max	Min
1.	7.3.	branje	44	22	52	22	2.36	52	22	2.36	2.36	2.36
	Opis: Fenton: SW Metrics (1.pog) (minut na stran)											
2.	7.3.	prog.	100	100								
	Opis: Prijava v testni portal, uporaba (minut na ekvivalent vrstice kode)											
3.	9.3.	branje	31	13	80	13	6.15	132	35	3.77	6.15	2.36
	Opis: Ponovitev snovi pret.tedna (zvezek)											
4.	9.3.	branje	87	23	97	23	4.22	229	58	3.95	6.15	2.36
	Opis: Fenton: SW Metrics (2.pog)											

PSP: Upravljanje s časom

Ključni elementi pri upravljanju s časom:

1. Odločite se, kako želite porabiti svoj čas
2. Določite porabo razpoložljivega časa za predvidene aktivnosti
3. Spremljajte, kako porabljate čas v primerjavi z predvideno porabo časa
4. Odločite se, kako spremeniti aktivnosti v odvisnosti od predvidenega časa

PSP: Upravljanje s časom

1. Odločite se, kako želite porabiti svoj čas

a) Določite kategorije svojih aktivnosti
- preverite že določene kategorije aktivnosti

b) Zbirajte podatke o porabljenem času po aktivnostih
- tabela tedenske porabe časa po aktivnostih

c) Ovrednotite svojo porabo časa
- prioriteta aktivnosti?
- uravnoteženost med delom, zabavo in počitkom
- rezerva

PSP: Upravljanje s časom

2. Določite porabo razpoložljivega časa (za predvidene aktivnosti)

a) Plan, kako želimo porabiti razpoložljiv čas
- glede na zbrane podatke in predvidene zahteve
- Primer – 1. korak:

Kategorija	Predviden čas:	Dejanski čas:
Predavanja	135	
Vaje (avd. in lab.)	150	
Branje	90	
Program	60	
Račun	30	
SKUPAJ:	465	

Ime: Andrej Novak
Namen: Študij predmeta RPS2

Datum začetka: 16.05.2005
Zap.št. tedna: 10. teden

SEŠTEVEK ČASOV TEKOČEGA TEDNA

Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Internet	SKUPAJ
SKUPAJ	1350	450	1050	1092	212	610	373	5137
POVPR.	135	45	105	109	21	61	37	514
MAX.	135	45	105	269	182	375	128	850
MIN.	135	45	105					312

Skupno število tednov: 10

Pristop:

- presodi, kaj je potrebno in kaj ne
- presodi, kaj je izvedljivo in kaj ne
- prioriteta >> učinkovitost
- dodatne aktivnosti – dodaten čas
- problem dodatnega časa:
  - časovna rezerva
  - prerazporeditev aktivnosti

Ključno pravilo:

- uravnoteženost
- dolgotrajno izvajanje
- moja izkušnja ...

Primer – 2. korak:

- prilagoditev za izpitni teden

Kategorija	Predviden čas:	Dejanski čas:
Predavanja	135	
Vaje (avd. in lab.)	150	
Branje	~~60~~ 30	
Program	~~60~~ 0	
Račun	~~30~~ 90	
Priprava izpita	150	
SKUPAJ:	~~465~~ 555	

3. Spremljajte, kako porabljate predviden čas

- najtežji del upravljanja s časom !
- redne : izredne aktivnosti
- pomaga, če določimo fiksna pravila za porabo časa
- predvideti zgolj čas po aktivnostih je premalo - urnik po dnevih
- uvedemo dodatne aktivnosti
- več vrst urnikov – recimo:
  - običajni teden: predavanja + običajen študij
  - teden oddaje (dela) seminarske naloge
  - izpitni teden: posebne priprave za izpit poleg ostalega

Primera : običajen teden, izpitni teden (predavanja + izpit v petek)

PSP: Upravljanje s časom

Ime: Andrej Novak
Vrsta: Običajen teden

Datum začetka: 16.05.2005
Zap.št.tedna: 10. teden

Dan\Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program		SKUPAJ
PON,16.5.			105	30				135
TOR,17.5.					30	30		60
SRE,18.5.		45		30				75
ČET,19.5.	135					30		165
PET,20.5.				30				30
SOB,21.5.								
NED,22.5.								
SKUPAJ	135	45	105	90	30	60		465

PSP: Upravljanje s časom

Ime: Andrej Novak
Vrsta: Izpitni teden

Datum začetka: 06.06.2005
Zap.št.tedna: 15. teden

Dan\Kat	Pred.	A.vaje	L.vaje	Branje	Račun	Program	Izpit	SKUPAJ
PON,6.6.			105	30				135
TOR,7.6.					60		60	120
SRE,8.6.		45			30		60	135
ČET,9.6.	135						30	165
PET,10.6.							xxx	
SOB,11.6.								
NED,12.6.								
SKUPAJ	135	45	105	30	90		150	555

PSP: Upravljanje s časom

Tri skupine opravil:

1. **Obvezna opravila, za katera je čas izvedbe fiksno določen:**
- najvišja prioriteta
- primeri: predavanja, vaje
2. **Nujna opravila, za katera je čas izvedbe spremenljiv:**
- lahko jih optimalno razporedimo
- primeri: branje, programiranje
3. **Ostala opravila:**
- opravila, ki ne služijo izbranemu namenu
- izpopolnitev preostalega časa
- primeri: prehrana, spanje, zabava, telovadba

PSP: Upravljanje s časom

Skupine opravil - določanje urnika:

1. **Obvezna opravila:** natanko določeni termini
2. **Nujna opravila:** uravnoteženo razporedimo z vidika drugih opravil
3. **Ostala opravila:** fiksni ali spremenljivi termini, v skladu z določenimi kriteriji oziroma pravili

Urnik za vse aktivnosti tedna daje celovito sliko:

- za celotno zalogo časa v tednu (7 * 24 * 60 = 10080 minut)
- namesto beležke porabe časa
- vidimo, kako lahko prerazporedimo opravila
- ostala opravila določajo rezervno zalogo časa

PSP: Upravljanje s časom

Izhodišča za razporejanje opravil:

- razporedi opravila po prioriteti
- ugotovi, če imajo določena opravila rok, do katerega morajo biti končana
- razporedi, katera opravila morajo biti izvedena pred drugimi
- ugotovi, katera opravila morajo biti izvedena naenkrat
- določi vrstni red opravil, ki bi jih raje prej izvedel
- ne odlašaj z nepriljubljenimi opravili

PSP: Upravljanje s časom

Ime: Andrej Novak — Datum začetka: 06.06.2005

Sk.aktiv.	RPS2	ORG	MUI	RK	RA	Hr./Sp.	Ostalo	SKUPAJ
FIK.OPR.								
Pr.+Vaje	270	180	270	270	135			1125
SPR.OPR.								
Branje	180	90	120	120	30			540
Naloge	120	90	180	30	240			660
OST.OPR.								
Hrana						1260		1260
Spanje						3150		3150
Zabava*							3345	3345
SKUPAJ	570	360	570	420	405	4410	3345	10080

PSP: Upravljanje s časom

Nasveti:

- planiraj porabo časa realistično
- planiraj dolgoročno
- predvidi dovolj časa za spanje, prehrano, zabavo
- urnik naj bo enostaven in odraža jasna pravila
- pošteno spremljaj dejansko porabo časa
- po potrebi spremeni predviden čas za opravila
- šele po določenem času bo urnik zares uporaben

In še: dejansko porabi za opravila toliko časa, kot si predvidel

- predvidena poraba časa jasno odraža zastavljene prioritete
- urnik mora motivirati

PSP: Upravljanje zavez

Zaveza (ang. commitment):

Iz kakršnega koli razloga smo se zavezali, da bomo nekaj za nekoga opravili

Dve vrste zavez:

a) Pogodbena zaveza:

- odraža razmerje zahtev in pričakovanj med strankama
- sporazum (dogovor) med dvema ali več strankami pred izvedbo opravila
- določa čas in plačilo za izvedeno opravilo
- primer: pogodbena zaveza med A in B

b) Osebna zaveza:

- vsebuje osebno privolitev - je prostovoljna
- nekaj, kar zmoremo in hočemo izvesti
- osebni, neformalen dogovor: tudi tu je znan čas izvedbe in plačilo
- primer: napredovati v naslednji letnik

PSP: Upravljanje zavez

Prava zaveza je:

- pogodbena in osebna hkrati
- vsebuje tako osebni kot eksplicitno definiran dogovor
- dogovor je med dvema ali več strankami
- **obsega:**
  - kaj bo narejeno
  - kriterije, ki določajo, kdaj je opravilo dokončano
  - kdo bo izvedel opravilo
  - do kdaj bo opravilo dokončano
  - kaj bo povračilo (plačilo) za opravljeno delo
  - kdo bo plačnik

PSP: Upravljanje zavez

Ustrezna zaveza določa tudi:

- odgovornost in
- učinkovito upravljanje

Da bi to dosegli moramo:

- natanko analizirati opravilo pred sklenitvijo dogovora (zaveze)
- izvedba mora biti definirana in formalizirana v obliki plana izvedbe
- dogovor mora biti dokumentiran
- če zaveze ni mogoče izpolniti, je treba pravočasno o tem obvestiti drugo stranko in kar najbolj zmanjšati negativne posledice

Primer: planiranje vodstva : planiranje po PSP principih

PSP: Upravljanje zavez

Primer:

Sodelovanje študenta pri laboratorijskem projektu

1. Obisk študenta pri predstojniku laboratorija (natančen opis opravila)
2. Odločitev za sodelovanje (zmožnost in želja)
3. Študent prerazporedi aktivnosti, da lahko nameni dovolj tedenskega časa za tedensko sodelovanje pri delu
4. Študent določi urnik (preveri zmožnost)
5. Študent drugič obišče predstojnika (dogovor o urniku in plačilu)
6. Študent in predstojnik podpišeta dogovor
7. Študent začne z delom

PSP: Upravljanje s časom

Sk.aktiv.	RPS2	ORG	MUI	RK	RA	Hr./Sp.	Ostalo	SKUPAJ
FIK.OPR.								
Pr.+Vaje	270	180	270	270	135			1125
Lab.delo							600	600
SPR.OPR.								
Branje	180	90	120	120	30			540
Naloge	120	90	180	30	240			660
OST.OPR.								
Hrana						1260		1260
Spanje						3150		3150
Zabava*							2745	2745
SKUPAJ	570	360	570	420	405	4410	3345	10080

PSP: Upravljanje zavez

Ime: Andrej Novak
Urnik fiksnih opravil

Datum začetka: 16.05.2005
Zap.št.tedna: 10. teden

Dan\Kat	RPS2	ORG	MUI	RK	RA	Lab.delo	Ostalo	SKUPAJ
PON,16.5.	10.00 – 12.00*	15.00 – 17.00*				12.00 – 15.00		360
TOR,17.5.			12.00 – 15.00*		15.00 – 16.00*	8.00 – 12.00		420
SRE,18.5.	11.00 – 12.00*		8.00 – 11.00*	12.00 – 15.00*				315
ČET,19.5.	7.00 – 10.00*	13.00 – 15.00*		10.00 – 13.00*				360
PET,20.5.					9.00 – 11.00*	11.00 – 14.00		270
SOB,21.5.								
NED,22.5.								
SKUPAJ	270	180	270	270	135	600		1725

PSP: Upravljanje zavez

Problem neizpolnjenih zavez:

- narava nekaterih opravil je nepredvidljiva (razvoj PO)
- plan, ki upošteva vse možne “nepredvidljivosti’, je nerealen
- pomembna je politika, kako urediti neizpolnjene zaveze:
  - ne odlašaj - pravočasno obvesti stranko (naročnika)
  - poskrbi, da so posledice za naročnika čim manjše - tipično:
    - če mogoče, prestavi rok za konkretno opravilo in predvidi, kako boš nadoknadil čas
    - dokončaj osrednji del opravila (funkcionalno najpomambnejše funkcije), predvidi kdaj bo delo v celoti dokončano
- minimiziraj vpliv svojih negativnih posledic (ugled, plačilo)
- problem iskanja krivca
- **Vendar: ne obupaj prezgodaj !** (alternative, zunanja pomoč, ...)

PSP: Upravljanje zavez

Upravljanje zavez omogoča, da:

- pravočasno izpolnimo vse zaveze
- ob krizah optimalno rešimo težave

Posledice slabega upravljanja zavez:

- izvedba opravila zahteva več časa, kot smo ga predvideli
- zaveze ne uspemo izpolniti
- napačno določene prioritete zavez
- slaba kakovost izvršenih opravil
- izguba zaupanja
- izguba zaupanja vase (v svojo presojo)
- izguba denarja

PSP: Upravljanje zavez

Primer seznama zavez:

Ime: Andrej Novak **Datum začetka: 16.05.2005**

Seznam zavez

Datum	Zaveza	Komu?	Ure	Rok	Plačilo
P, S, Č	Predavanja + vaje RPS2	Pred. Rožanc	6*		Ocena
P, Č	Predavanja + vaje ORG	Doc. Jager	4*		Ocena
T, S	Predavanja + vaje MUI	Prof. Kononenko	6*		Ocena
S, Č	Predavanja + vaje RK	Doc. Vidmar	6*		Ocena
T, Pe	Predavanja + vaje RA	Prof. Leonardis	3*		Ocena
P, T, Pe	Delo v Laboratoriju	Prof. XY	10		Denar
7.6.	Izpit RPS2	Pred. Rožanc	15	6.6.	Ocena

PSP: Upravljanje zavez

Nasveti v zvezi z izpolnjevanjem zavez v razvoju PO:

- če zamujamo z izvedbo trenutnega opravila, bodo kasnila tudi vsa sledeča opravila, dokler nečesa korenito ne spremenimo
- več truda običajno ne reši težave, sploh pa ne na daljši rok
- če ne znaš določiti, kakšen je trenuten status izvedbe opravila (koliko še manjka do konca), si skoraj gotovo v težavah
- če se pri izpolnjevanju zahtev zanašaš na srečo, zaveze skoraj gotovo ne boš izpolnil
- če so ocene napačne, so zelo verjetno prenizke in ne previsoke
- vsaka sprememba zahteva več dela (kot ga predvidimo)
- agresivno težiti k izpolnitvi zahtev je nujno potrebno, ne pa tudi nujno dovolj
- (majhnih) začetnih napak ne more rešiti (veliko) truda na koncu

PSP: Urniki

Urnik:

- nujen, ko želimo istočasno izvajati veliko opravil

Tipično:

a) več (manjših) opravil, ki se vlečejo daljši čas (predavanja)

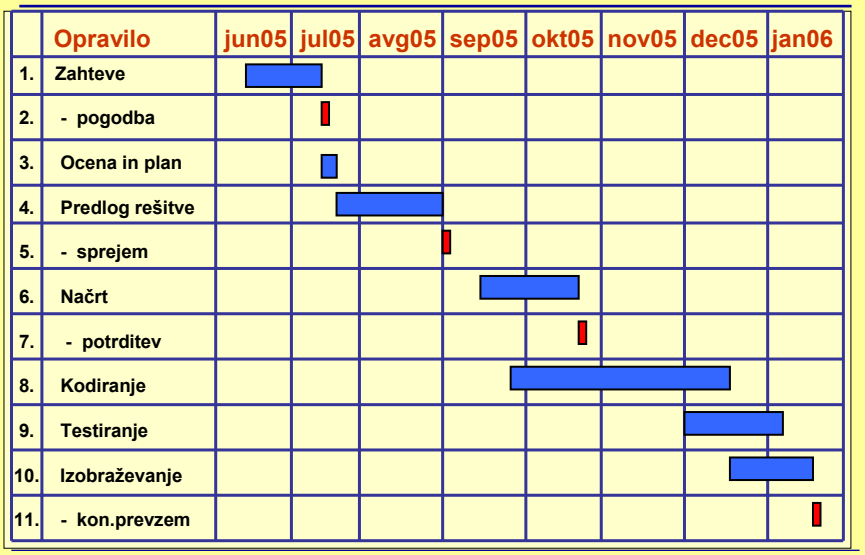
b) večje opravilo, ki ga delimo na veliko majhnih (pod)opravil

- zaporedno izvajanje opravil sešteva učinke (zamude)
- urnik zasnujemo in ga postopoma dopolnjujemo

Primer: izvedba osebnega projekta : študij predmeta

Urnik razvoja PO: Ganttov diagram

PSP: Urniki



PSP: Urniki

Izdelava urnika projekta:

1. natančno analiziraj delo in določi vsa opravila, ki ga sestavljajo
2. določi obseg posameznih opravil in količino dela zanje
3. nariši Ganttov diagram, ki vključuje vsa opravila (začetke, konce)

Dodatno za urnike posameznih izvajalcev še:

1. preveri, da vsak izvajalec pozna opravila, ki jih mora opraviti
2. pridobi potrjene roke za izvedbo teh opravil
3. določi medsebojen vpliv posameznih opravil
4. dokumentiraj medsebojen vpliv
5. ponovno preveri urnike in medsebojne vplive z vsemi udeleženci ted odpravi konflikte
6. posebej preveri celoten urnik z vidika popolnosti

PSP: Urniki

Mejniki (checkpoints, milestones):

- projekte razdelimo na manjše dele – ti predstavljajo “merljive dele” projekta
- zaključek takega dela je mejnik

Mejnik: objektivno jasno določljiva točka pri razvoju projekta

- jasna in nedvoumna definicija
- ključnega pomena za planiranje

Tipični mejniki: podpisana pogodba, oddana seminarka naloga, dokumentiran plan, uspešno preveden program, stestiran program ...

Neprimerni mejniki: 90% zaključeno testiranje, zasnovan program, pregledana knjiga, stanje pred opravljanjem izpita (lahko da ali ne)

PSP: Urniki

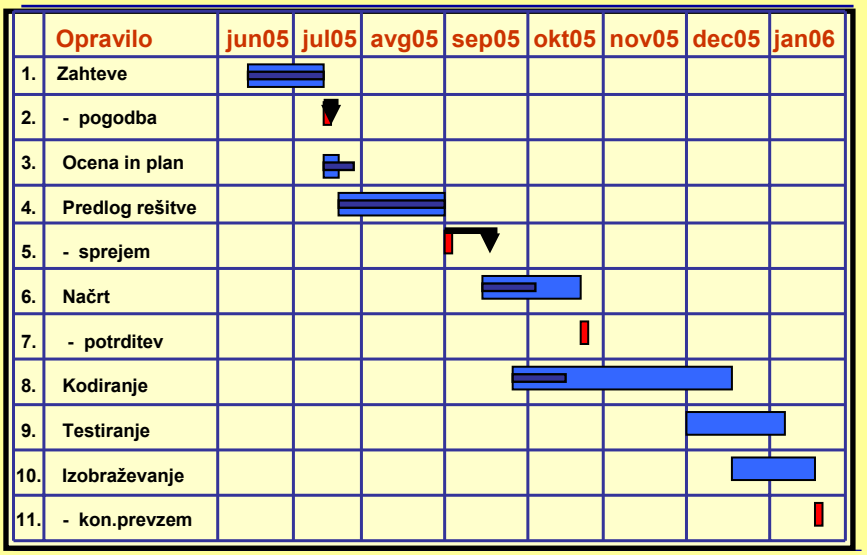
Nasveti za določanje mejnikov (PSP):

- določimo več mejnikov za neko opravilo na teden
- mejnik naj sledi primerni količini dela (recimo vsakih 5 ur)
- tudi manj intenzivno delo naj ima vsaj en mejnik na teden
- problem krajših rutinskih opravil
- če sodelujemo v skupini (medsebojna odvisnost), moramo mejnike postaviti na konec smiselnih delov opravil
- pomen mejnikov za spremljanje dela v projektni skupini

Spremljanje izvajanja projektnega plana

- zanima nas status projekta: teče po planu, prehiteva ali zaostaja?
- nepogrešljivo za večje projekte z roki izvedbe
- omogoča pravočasno odkrivanje problemov
- primer: Ganttov diagram omogoča planiranje in spremljanje

PSP: Urniki



PSP: Urniki

Problem spremljanja izvajanja projektnega plana

- nenatančno spremljanje izvajanja zavaja
- problem spreminjanja originalnega plana

Do tega ne bo prišlo, če:

- ustrezno definiramo in dokumentiramo mejnike
- ne spreminjamo plana, dokler ne končamo vseh opravil
- tudi ko ugotovimo odstopanja, ne spremenimo plana
- če ga le spremenimo, spremembe vnašamo z drugo barvo
- če plana nismo definirali sami (skupina) preverjamo originalnost
- ugotovitve smiselno vgradimo v nov plan

Vedno spremljamo izvajanje glede na originalen plan!

PSP: Urniki

Spremljanje deleža opravljenega dela (earned value)

- jasen vpogled v trenutni status dela
- za določena opravila enostavno, za druga ne
- smiselne časovne enote (tedni, dnevi, ure)

Primer: delež opravljenega dela po 10 dneh pisanja diplomske naloge

Predpostavke:

- spremljamo samo dejansko pisanje (ostalo smo že zaključili)
- diplomsko nalogo pišemo zaporedoma (poglavje za poglavjem, stran za stranjo)
- na podlagi že opravljenega dela smo načrtovali strukturo diplomske naloge in obseg posameznih poglavij

Plan:

- vsako stran pišemo 2 uri
- vsak dan efektivno delamo 8 ur

PSP: Urniki

Plan je zapisan v obliki tabele:

- Poglavje
- Opis
- Planiran čas
- Planiran delež dela
  - delež enote
  - kumulativni delež
- Časovni interval
  - planiran dan
  - dejanski dan
- Delež opravljenega dela

PSP: Urniki

Pogl.	Opis	Plan (min)	Plan. delež		Časovni interval		Dosežen delež
			enota	skup.	plan	dej.	
PI	Plan strukture	180	1.887	1.887	1	1	1.887
1. (5)	Uvod	600	6.289	8.176	2	1	6.289
2.(10)	Opis problema	1200	12.579	20.755	5	4	12.579
3.(15)	Opis rešitve	1800	18.868	39.623	8	6	18.868
4.(15)	Prak.izvedba	1800	18.868	58.491	12	10	18.868
5.(8)	Zaključek	960	10.063	68.554	14		
P	Pop. besedila	1200	12.579	81.133	17		
K (5)	Kazala,naslov	600	6.289	87.421	18		
B	Ponovno branje	1200	12.579	100.000	20		
S (53)	Skupaj naloga	9540	100.000				

PSP: Urniki

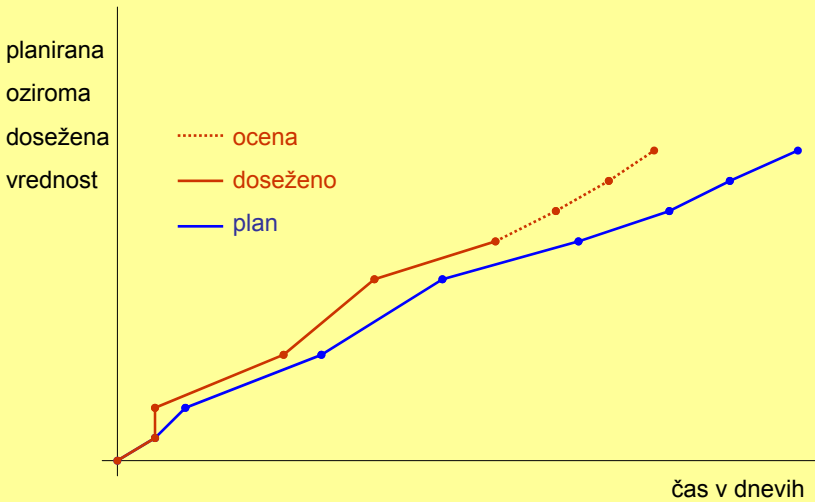
Dan	Planirane vrednosti	Dosežen delež	Ocena dela za naprej
1.	1.887	8.176	
2.	8.176	8.176	
3.	8.176	8.176	
7.	20.755	39.623	
8.	39.623	39.623	
9.	39.623	39.623	
10.	39.623	58.491	
11.	39.623		58.491
16.	68.554		81.133
17.	81.133		100.000
20.	100.000		

PSP: Urniki

Primer:

- problem smiselnih časovnih enot
- mejniki in natančnost časovnega plana
- časovni zamiki v planu
- problem spreminjanja plana
- prilagajanje plana svojim potrebam
- grafična predstavitev dosežene vrednosti dela

PSP: Urniki



PSP: Projektni plan

Projektni plan:

- ključni del projekta
- opisuje delo, ki ga moramo opraviti
- definira vsa večja opravila
- ocenjuje potreben čas in vire
- okvirno opredeljuje način upravljanja in nadzora projekta

Velja:

- plan mora biti obvezno ustrezno dokumentiran
- obsežnejši problem – večji pomen plana
- nujna vsaj ocena potrebnega časa in velikost izdelkov

PSP: Projektni plan

Projektni plan z našega vidika:

- Obrazec
- Začeli bomo z enostavno različico, ki jo bomo postopoma dopolnjevali
- **Namen:** planiranje pisanja manjših programov
- **Vsebina:**
  - osnovni podatki o opravilu - programu
  - velikost izdelka
  - čas razvoja
  - skupne značilnosti (dosedanjega dela)

Dva vidika:

- planirane vrednosti : izmerjenim vrednostim
- največje, najmanjše in povprečne vrednosti

PSP: Projektni plan

PROJEKTNI PLAN

Izvajalec: _____ **Datum:** _____
Program: _____ **Zap.št:** _____
Vodja: _____ **Pr.jezik:** _____

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:		
LOC/uro:		

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**
Skupaj (nove+spr.):		
Največja velikost:		
Najmanjša velikost:		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**
Skupaj:		
Največji čas:		
Najmanjši čas:		

PSP: Projektni plan

Izpolnjevanje projektnega plana:

- planirane vrednosti vnesemo pred opravilom (programiranjem)
- dejanske vrednosti vnesemo ob zaključku opravila
- vnos vrednosti si bomo ogledali po delih

OSNOVNI PODATKI:

- izvajalec opravila (študent)
- ime (namen, opis) programa
- vodja: nadzornik, pregledovalec, vodja projektne skupine, učitelj
- datum začetka izvajanja opravila
- zaporedna številka opravila
- programski jezik (programsko orodje)

PSP: Projektni plan

SKUPAJ (skupne značilnosti dosedanjega dela):

- razmerja za planiranje in dejansko dosežena razmerja
- planirane vrednosti vnesemo na podlagi izkušenj ali ocene

Min/LOC: tipično povprečno razmerje iz beležke opravil (za programiranje)

- upoštevamo značilnosti
- vrednost naj bo med največjim in najmanjšim razmerjem

LOC/uro:

- obratna vrednost, pomaga pri planiranju
- izračun:

[LOC/uro] = 60 / [Min/LOC]

Recimo: [Min/LOC] = 8.78 [LOC/uro] = 60 / 8.78 = 6.83

PSP: Projektni plan

Primer beležke opravil

Ime: Andrej Novak **Datum začetka: 11.03.2005**

# Opr	Datum	Vrsta	Ocena		Izmerjeno			Doslej				
			Čas	Enot	Čas	Enot	Raz	Čas	Enot	Raz	Max	Min
14.	11.3.	prog	60	20	102	14	7.29	102	14	7.29	7.29	7.29
	Opis: Program 3 (minut na LOC)											
15.	11.3.	prog	182	25	204	23	8.87	306	37	8.27	8.87	7.29
	Opis: Program 4											
16.	13.3.	branje	50	13	45	13	3.46	332	84	3.95	6.15	2.36
	Opis: Ponovitev snovi pret.tedna (zvezek)											
17.	14.3.	prog	177	20	300	32	9.63	606	69	8.78	9.37	7.29
	Opis: Program 5											

PSP: Projektni plan

VELIKOST (programa v LOC):

- spremljamo velikost novih in spremenjenih delov programa
- upoštevamo standaren način merjenja LOC
- planirane vrednosti vnesemo na podlagi izkušenj ali ocene

Skupaj (nove in spremenjene vrstice):

- za oceno velikosti uporabimo pristop z opredelitvijo značilnosti programa
- upoštevamo značilnosti – posebnosti programa

Največja / najmanjša velikost:

- zapišemo tudi ti vrednosti iz ocene velikosti
- vrednosti omogočata opredelitev območja možnih velikosti
- koristno za oceno časa

PSP: Projektni plan

Ocena velikosti za program 6

Vrsta	Min/LOC	Zahtev.	Značilnost	Min_{LOC}	Ave_{LOC}	Max_{LOC}
Zanka						
1x	6.27	enost.	FOR ali WHILE	7	8	16
1x	7.33	sred.	REPEAT	8	10	22
Pogoj						
1x	6.90	sred.	IF	7	10	13
Logika						
3x	7.53	enost.	izračun	5	7	12
OCENA				37	49	87

Komentar: **program za izračun povprečne pozitivne ocene vseh študentov PO na UNI RI - dvodimenzionalna tabela (študent x letnik)**

PSP: Projektni plan

ČAS RAZVOJA (programa v minutah):

- ocenjujemo in merimo čas, ki je potreben za izvedbo celotnega opravila
- planirane vrednosti izračunamo iz podatkov o razmerjih in velikosti programa
- dejanske vrednosti čim natančneje izmerimo

Skupaj :

- planirano vrednost izračunamo na naslednji način:

[skupaj čas] =[skupaj velikost] * [Min/LOC]

Recimo: [skupaj velikost] = 49
[Min/LOC] = 8.78

[skupaj čas] = 49 * 8.78 = 430.22 = 430 minut

(za primerjavo - ocena z upoštevanjem značilnosti programa: 350.59)

PSP: Projektni plan

Največji / najmanjši čas:

- uporabimo enak pristop za največjo / najmanjšo oceno velikosti

[največji čas] =[največja velikost] * [Min/LOC]

[najmanjši čas] =[najmanjša velikost] * [Min/LOC]

Recimo: [največja velikost] = 87
[najmanjša velikost] = 37
[Min/LOC] = 8.78

[največji čas] = 87 * 8.78 = 763.86 = 764 minut
[najmanjši čas] = 37 * 8.78 = 324.86 = 325 minut

(za primerjavo - oceni z upoštevanjem značilnosti programa: 263,78 oziroma 619.94)

PSP: Projektni plan

PROJEKTNI PLAN (verzija 1)

Izvajalec:	**ANDREJ NOVAK**	**Datum:**	**17. 3. 2005**
Program:	**PROGRAM 6 (pov. oc. štud PO)**	**Zap.št:**	**18**
Vodja:	**PRED. ROŽANC**	**Pr.jezik:**	**JAVA**

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:	**8.78**	
LOC/Uro:	**6.83**	

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**
Skupaj (nove+spr.):	**49**	
Največja velikost:	**87**	
Najmanjša velikost:	**37**	

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**
Skupaj:	**430**	
Največji čas:	**763**	
Najmanjši čas:	**325**	

PSP: Projektni plan

Po končanem programiranju velja:

Izmerjena velikost programa : 62 LOC (standardnih)

Izmerjen čas: 512 minut

Razmerja med časom razvoja in velikostjo izračunamo in zapišemo:

[Min/LOC] =[izmerjen čas] / [izmerjena velikost]

torej: [Min/LOC] = 512 / 62 = 8.26

in

[LOC/Uro] = 60 / [Min/LOC]

torej: [LOC/Uro] = 60 / 8.26 = 7.26

PSP: Projektni plan

PROJEKTNI PLAN (verzija 1)

Izvajalec:	**ANDREJ NOVAK**	**Datum:**	**17. 3. 2005**
Program:	**PROGRAM 6 (pov. oc. študPO)**	**Zap.št:**	**18**
Vodja:	**PRED. ROŽANC**	**Pr.jezik:**	**JAVA**

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:	**8.78**	**8.26**
LOC/Uro:	**6.83**	**7.26**

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**
Skupaj (nove+spr.):	**49**	**62**
Največja velikost:	**87**	
Najmanjša velikost:	**37**	

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**
Skupaj:	**430**	**512**
Največji čas:	**763**	
Najmanjši čas:	**325**	

PSP: Projektni plan

Vrednosti vnesemo tudi v:

- beležko porabe časa (tabele skupne porabe časa)
- beležko opravil

Izkušnje z uporabo projektnega plana:

- Na začetku bodo ocene zelo nenatančne
- Če pravilno planiramo, bo približno pol prevelikih in pol premajhnih ocen
- Ocene bodo natančnejše z več izkušnjami
- Ključno za uspešno planiranje:
  - prepoznati pomembne značilnosti programa, ki vplivajo na planiranje
  - občutek, kako ovrednotiti te značilnosti
- Plane je treba obvezno ustrezno dokumentirati
- Izvedene plane uporabljamo kot izhodišče za nadaljnja planiranja

Navodilo za izpolnjevanje projektnega plana – 1 (1/2)

Namen: Obrazec vsebuje planirane in dejansko izmerjene podatke o projektu v čitljivi in uporabni obliki

Glava: Vnesite:

- ime razvijalca in datum začetka
- naziv in zaporedno številko programa
- ime vodje
- programski jezik, v katerem bo napisan program

Min/LOC: Pred razvojem:

- vnesite planirano vrednost [Min/LOC]

Po razvoju:

- dejanski [Min/LOC] = [izmerjen čas] / [izmerjena velikost]

LOC/Uro: Pred razvojem:

- planiran [LOC/Uro] = 60 / [Min/LOC]

Po razvoju:

- dejanski [LOC/Uro] = 60 / [Min/LOC]

Navodilo za izpolnjevanje projektnega plana – 1 (2/2)

Velikost (LOC): Pred razvojem:

- vnesite planirane vrednosti za skupno, največjo in najmanjšo velikost nove in spremenjene kode

Po razvoju:

- seštejte dejanske vrstice nove in spremenjene kode

Čas razvoja (min): Pred razvojem:

- določite planirane vrednosti časa razvoja po naslednjih formulah:
  [planiran skupni čas] = [planirana skupna velikost] * [Min/LOC]
  [planiran največji čas] = [planirana največja velikost] * [Min/LOC]
  [planiran najmanjši čas] = [planirana najmanjša velikost] * [Min/LOC]

Po razvoju:

- izmerite dejanski čas razvoja (s pomočjo Beležke porabe časa)

PSP: Proces razvoja programske opreme

Proces razvoja programske opreme:

- Definira postopek (korake), kako lahko izvedemo določeno delo
- Odgovarja na vprašanja: kako, kdo, kdaj in za kakšno ceno
- Ima jasno hierarhično strukturo:
  - celoten proces je običajno sestavljen iz več delov - faz
  - običajno so faze sestavljene iz manjših delov – korakov ali opravil
  - proces ima lahko eno ali več faz
  - faza ima lahko eno ali več opravil (korakov)
  - vsak del je defineran tudi z vhodi (viri) in izhodi (rezultati)
- Če je proces ustrezno dokumentiran, je **definiran proces**
- Obstajajo modeli, ki določajo, kaj mora vsebovati definicija procesa

PSP: Proces razvoja programske opreme

Več formalnih definicij:

Definicija 1: *Proces razvoja PO je sistem vseh opravil in potrebnih orodij, standardov, metod, postopkov in praks, ki so udeleženi pri proizvodnji in razvoju izdelkov PO skozi celoten življenjski cikel PO [Rozum 2000].*

Definicija 2: *Proces razvoja PO je množica aktivnosti, metod, praks in transformacij, ki jih ljudje uporabljajo pri razvoju in vzdrževanju PO in z njo povezanih izdelkov (projektnih planov, načrtov, kode, testnih primerov, uporabniških priročnikov)[Paulk 1993].*

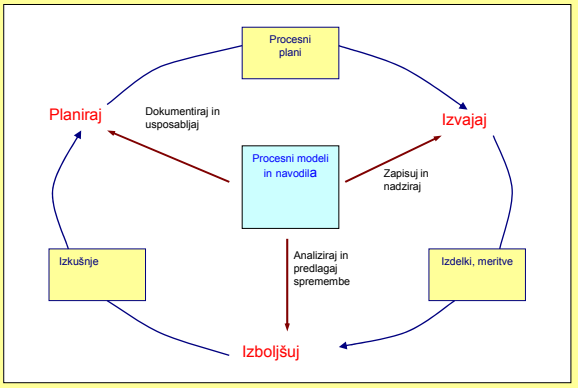
Procese razvoja PO lahko definiramo enako kot vsak izdelek PO:

- ugotovimo potrebe,
- določimo cilje in kriterije kakovosti,
- zasnujemo nov proces na podlagi obstoječega stanja,
- proces preverimo in
- ga postopoma izboljšujemo.

PSP: Proces razvoja programske opreme

Tri faze modeliranja procesa:

a) **Planiraj:** zasnova osnove procesa razvoja PO glede na obstoječe stanje;
b) **Izvajaj:** spremljanje stanja procesa;
c) **Izboljšuj:** izvedba potrebnih sprememb, da dosežemo pričakovano stanje.

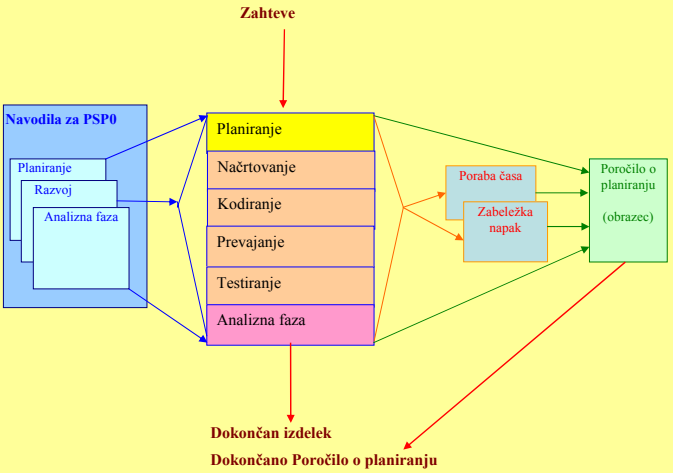


PSP: Proces razvoja programske opreme

Z vidika PSP:

- proces določamo za inženirja za razvoj PO (programerja, študenta)
  - začetni proces: samo značilnosti običajnega pristopa pri delu (PSP0)
  - proces postopoma nadgradimo z:
    - merjenjem (PSP0.1),
    - planiranjem (PSP1),
    - zagotavljanjem kakovosti (PSP2)
    - obvladovanjem večjih opravil (PSP3)
  - proces je definiran jasno in enostavno:
    - z navodili,
    - obrazci,
    - beležkami in
    - standardi

Struktura osebnega procesa



Definiran osebni proces razvoja PO (PSP0):

Namen

Vhod

a) planiranje
b) načrtovanje
c) kodiranje
d) prevajanje
e) testiranje
f) analizna faza

Izhod

PSP: Proces razvoja programske opreme

Namen: postopek razvoja majhnih programov

Vhodni pogoj:

- opis problema
- Projektni plan
- zbrani zgodovinski podatki o dejanski porabi časa in velikosti programov
- Beležka porabe časa

1. Planiranje

- pridobi opis značilnosti programa
- oceni velikost programa (pričakovano, največjo in najmanjšo)
- določi [Min/LOC]
- določi čas razvoja (pričakovan, največji in najmanjši)
- vnesi planirane vrednosti v Projektni plan
- vnesi čas planiranje v Beležko porabe časa

PSP: Proces razvoja programske opreme

2. Načrtovanje

- načrtuj program
- opiši načrt programa v predvidenem formatu
- vnesi čas načrtovanja v Beležko porabe časa

3. Kodiranje

- Implementiraj načrt (s programom)
- uporabi standarden format pri kodiranju
- vnesi čas kodiranja v Beležko porabe časa

4. Prevajanje

- prevedi program
- odpravi vse odkrite napake
- vnesi čas prevajanja v Beležko porabe časa

5. Testiranje

- Testiraj program
- odpravi vse odkrite napake
- vnesi čas testiranja v Beležko porabe časa

6. Analizna faza

- do konca izpolni Projektni plan (dejanski čas razvoja, velikost, razmerja)
- vnesi čas analize v Beležko porabe časa

Izhodni pogoj

- popolno testiran program (rezultat)
- popolno dokumentiran načrt
- celotna koda programa
- izpolnjen Projektni plan
- vnešeni ustrezni podatki v Beležko porabe časa

PSP: Proces razvoja programske opreme

Mejniki:

- namesto nekaj mejnikov imamo mejnik ob koncu vsake faze
- vsaka faza procesa ima natanko predpisane rezultate - izdelke
- končani izdelki so merljiv mejnik

Dopolnimo projektni plan

- natančneje spremljamo **čas razvoja**
- poleg celotnega časa spremljamo tudi porabo časa za vsako fazo posebej
- spremljamo tudi skupno dosedanjo porabo časa po fazah ter delež skupnega časa v posameznih fazah
- tudi z vidika velikosti programa spremljamo skupno velikost vseh programov doslej
- nove rubrike vnesemo na ustrezna mesta v projektnem planu (verzija 2)

PSP: Proces razvoja programske opreme

PROJEKTNI PLAN (verzija 2)

Izvajalec: _____ **Datum:** _____
Program: _____ **Zap.št:** _____
Vodja: _____ **Pr.jezik:** _____

SKUPAJ:	**Plan:**	**Dej.vrednost:**		
Min/LOC:				
LOC/Uro:				
VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	
Skupaj (nove+spr.):				
Največja velikost:				
Najmanjša velikost:				
ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				
Načrtovanje				
Kodiranje				
Prevajanje				
Testiranje				
Analiza				
Skupaj:				
Največji čas:				
Najmanjši čas:				

Navodilo za izpolnjevanje projektnega plana – 2 (1/3)

Namen: Obrazec vsebuje planirane in dejansko izmerjene podatke o projektu v čitljivi in uporabni obliki

Glava: Vnesite:

- ime razvijalca in datum začetka
- naziv in zaporedno številko programa
- ime vodje in programski jezik, v katerem bo napisan program

Min/LOC: Pred razvojem:

- vnesite planirano vrednost [Min/LOC]. **Uporabite skupno vrednost Min/LOC iz zadnjega projektnega plana ali beležke opravil.**

Po razvoju:

- dejanski [Min/LOC] = [izmerjen čas] / [izmerjena velikost]

LOC/Uro: Pred razvojem:

- planiran [LOC/Uro] = 60 / [Min/LOC]

Po razvoju:

- dejanski [LOC/Uro] = 60 / [Min/LOC]

Navodilo za izpolnjevanje projektnega plana – 2 (2/3)

Velikost (LOC): Pred razvojem:

- vnesite planirane vrednosti za skupno, največjo in najmanjšo velikost nove in spremenjene kode

Po razvoju:

- seštejte dejanske vrstice nove in spremenjene kode
- **[doslej skupna velikost] = [dejanska velikost]+[doslej skupna velikost zadnjega projektnega plana]**

Čas razvoja (min): Pred razvojem:

- določite planirane vrednosti časa razvoja po naslednjih formulah:
  [planiran skupni čas] = [planirana skupna velikost] * [Min/LOC]
  [planiran največji čas] = [planirana največja velikost] * [Min/LOC]
  [planiran najmanjši čas] = [planirana najmanjša velikost] * [Min/LOC]

Navodilo za izpolnjevanje projektnega plana – 2 (3/3)

Čas razvoja (min): Pred razvojem (nadaljevanje):

- **poiščite [doslej %] vrednosti za vsako fazo posebej iz zadnjega projektnega plana**
- **izračunajte planirane čase razvoja na naslednji način:**
  [planiran čas faze] = [planiran skupni čas] * [doslej % faze] / 100

Po razvoju:

- izmerite dejanski čas razvoja **vsake faze in skupaj** (s pomočjo Beležke porabe časa)
- **[doslej skupni čas razvoja vsake faze in skupaj] =**
  [dejanski čas vsake faze in skupaj]+
  [doslej skupen čas razvoja vsake faze in skupaj iz zad.projektnega plana]
- **[doslej % skupni čas razvoja vsake faze] =**
  100 *
  [doslej skupen čas razvoja vsake faze] /
  [doslej skupni čas razvoja skupaj]

Primer:

- za planiranje potrebujemo podatke o preteklih projektih
- potrebujemo natančne podatke za vsako fazo posebej
- program številka 6 (povprečne ocene vseh študentov UNI RI smer PO) je prvi program, ki smo spremljali po dopolnjenem obrazcu
- Beležka porabe časa mora omogočati merjenje časa po posameznih fazah
- Predpostavimo, da:
  - smo pri planiranju porabo časa za posamezne faze ugibali
  - program ni zahteval posebnega načrtovanja, čeprav smo ga planirali

Projektni plan prikazuje stanje po zaključku projekta

- rdeča barva označuje nove vrednosti oziroma dodatke v primerjavi s predhodnim obrazcem

PSP: Proces razvoja programske opreme

PROJEKTNI PLAN (verzija 1)

Izvajalec:	ANDREJ NOVAK	**Datum:**	17. 3. 2005
Program:	PROGRAM 6 (pov. oc. štud PO)	**Zap.št:**	18
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:	8.78	8.26
LOC/Uro:	6.83	7.26

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**
Skupaj (nove+spr.):	49	62
Največja velikost:	87	
Najmanjša velikost:	37	

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**
Skupaj:	430	512
Največji čas:	763	
Najmanjši čas:	325	

PSP: Proces razvoja programske opreme

PROJEKTNI PLAN (verzija 2)

Izvajalec:	ANDREJ NOVAK	**Datum:**	17. 3. 2005
Program:	PROGRAM 6 (pov. oc. štud PO)	**Zap.št:**	18
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:	8.78	8.26
LOC/Uro:	6.83	7.26

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	49	62	62
Največja velikost:	87		
Najmanjša velikost:	37		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	30*	42	42	8.20
Načrtovanje	70*	0	0	0.00
Kodiranje	210*	314	314	61.33
Prevajanje	30*	44	44	8.60
Testiranje	50*	72	72	14.06
Analiza	40*	40	40	7.81
Skupaj:	430	512	512	100.00
Največji čas:	763			
Najmanjši čas:	325			

Program 7: poiskati in izpisati je treba študente z najvišjo povprečno oceno v vsakem letniku UNI RI smer PO

Planiranje pred začetkom razvoja:

- izpolnimo del obrazca v skladu z navodilom
- velikost ocenimo na podoben način kot pri programu 6
- razmerje porabljenega časa v posameznih fazah: ker program 6 nima načrtovanja, ga tudi tu ne planiramo

Vnos izmerjenih vrednosti po končanem razvoju:

- izmerimo končno velikost programa
- izmerimo čas v posameznih fazah in skupaj (beležka)
- izračunamo skupen čas v posameznih fazah doslej in razmerje skupnih časov posameznih faz
- določimo še Min/LOC in LOC/Uro

PROJEKTNI PLAN (verzija 2)

Izvajalec:	ANDREJ NOVAK	**Datum:**	22. 3. 2005
Program:	PROGRAM 7 (najboljši štud. po letnikih UNI RI - PO)	**Zap.št:**	20
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:	8.26	
LOC/Uro:	7.26	

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	73		
Največja velikost:	107		
Najmanjša velikost:	47		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	49			
Načrtovanje	0			
Kodiranje	370			
Prevajanje	52			
Testiranje	85			
Analiza	65			
Skupaj:	603			
Največji čas:	884			
Najmanjši čas:	388			

PSP: Proces razvoja programske opreme

PROJEKTNI PLAN (verzija 2)

Izvajalec:	ANDREJ NOVAK	**Datum:**	22 3. 2005
Program:	PROGRAM 7 (najboljši študi po letnikih UNI RI - PO)	**Zap.št:**	20
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**
Min/LOC:	8.26	5.43
LOC/Uro:	7.26	11.05

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	73	82	62+82=**144**
Največja velikost:	107		
Najmanjša velikost:	47		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	49	32	42+32=**74**	7.73
Načrtovanje	0	77	0+77=**77**	8.05
Kodiranje	370	240	314+240=**554**	57.89
Prevajanje	52	14	44+14=**58**	6.06
Testiranje	85	52	72+52=**124**	12.96
Analiza	65	30	40+30=**70**	7.31
Skupaj:	603	445	512+445=**957**	100.00
Največji čas:	884			
Najmanjši čas:	388			

PSP: Napake

Kakovost programske opreme:

- obsežno področje, ki zajema vrsto vidikov – recimo:
  - pravilno delovanje
  - odsotnost napak
  - uporabnost
  - učinkovitost
  - zanesljivost
  - robustnost
  - enostavna uporaba ...
- bistvo: definiramo jo z vidika pomembnih lastnosti za uporabnika
- vpliva na stroške, čas razvoja in zadovoljstvo uporabnika
- pomen jasnih zahtev za kakovost programske opreme
- pomen definiranega procesa za kakovost programske opreme
- modeli za zagotavljanje kakovosti

PSP: Napake

Z vidika posameznika - praktičen pristop

Kakovost PO = čim manj napak v programu = pravilno delovanje

Napaka - odpoved (defect): karšnakoli ovira programu, ki mu onemogoča, da bi popolnoma in učinkovito zadovoljil zahteve uporabnika:

- lahko jo objektivno definiramo, opišemo in štejemo
- lahko se zgodi v programu, načrtu, dokumentaciji, planu ...
- razlika med odpravljanjem napak, njihovimi vzroki in posledicami
- primeri: sintaksna napaka, napačen stavek, odvečen stavek, manjkajoč del programa (komentar!), nepopolna definicija ipd.
- napaka (defect): hrošč (bug)

Dva vidika obravnave napak v PSP-ju:

- **njihovo vnašanje v program**
- **odstranjevanje napak iz programa**

PSP: Napake

PSP na področju napak:

- **razlaga načine vnosa napak:**
  - vsi delamo napake (izkušeni programerji vsakih 7-9 LOC eno)
  - tudi po odstranjevanju napak le-te v izdelkih ostajajo
- **uvaja preventivne prijeme za zmanjšanje vnašanja napak:**
  - nujno: poznavanje prog. jezikov, načina dela, orodij, področja dela
  - standard za kodiranje
- **predlaga načine za učinkovito odkrivanje napak:**
  - razvijalčev proces razvoja programske opreme
  - pregledi programske kode
- **določa sistematične prijeme za odstranjevanje napak:**
  - beleženje in klasifikacija napak
  - spremljanje odstranjevanja napak

Ko ta vidik kakovosti obvladamo, se usmerimo k drugim (uporabnost, ...)

PSP: Napake

Kategorije napak:

- napake delimo na kategorije (glede na vzroke in posledice)
- smiselno število kategorij, nekatere kategorije ključne
- primer: Chillarage (IBM):
  - 10 kategorij
  - oznaka kategorij: 10, 20, 30, ..., 100
  - obsežna raziskava – smiselen predlog
  - različni programski jeziki – splošna uporabnost
  - sistem označevanja napak: od lažjih (10) proti zapletenim (100)
  - za začetek PSP predlaga to klasifikacijo
  - klasifikacijo (lahko) dopolnimo ali spremenimo, ko imamo nekaj izkušenj (vsaj 100 klasificiranih napak)

PSP: Napake

Kategorija	Naziv	Opis
10	Dokumentacija	komentarji, sporočila, opis delov programa
20	Sintaksa	tipkarske napake, ločila, format izpisa podatkov
30	Izgradnja	obvladovanje zahtev, moduli, knjižnice, verzije
40	Prirejanje	deklaracije, podvojena imena, veljavnost pravil, omejitve
50	Vmesnik	klici podprogramov, V/I, format podatkov
60	Preverjanje	obvladovanje napak, sporočila, pomankljivo preverjanje
70	Podatki	struktura, vsebina
80	Funkcije	logika, kazalci, zanke, rekurzija, izračuni, fun. napake
90	Sistem	konfiguracija, časovna razporeditev, spomin
100	Okolje	načrt, prevajanje, testiranje, podpora sistemu

PSP: Napake

Razumevanje napak:

- V ta namen moramo zbirati podatke o napakah
- Koraki:
  - zabeleži si vsako napako, ki jo najdeš v programu
  - o vsaki napaki shrani dovolj podatkov, da jo kasneje lahko razumeš
  - analiziraj podatke o napakah in raziskuj, katere vrste napak povzročajo največ težav
  - definiraj načine, kako bi preprečil oziroma lažje odkrival tovrstne napake

PSP definira Beležko napak, ki to omogoča

PSP: Napake

Struktura Beležke napak:

Glava:

- Razvijalec
- Vodja
- Datum
- Številka programa

Vsaka vrstica:

- Datum
- Številka napake
- Kategorija
- Faza vnosa
- Faza odprave (napake)
- Čas odprave (napake)
- Številka pop(ravljane) nap(ake, ki je povzročila to napako)
- Opis napake (lahko tudi mesto v programu, kjer se nahaja)

PSP: Napake

Ime: Andrej Novak
Vodja: Pred. Rožanc

Datum: 07.04.2005
Št.prog: 9

Datum	Št.nap.	Kategorija	Faza vnosa	Faza odprave	Čas odprave	Št.pop.nap.
8.4.	1	20	kodiranje	prevajanje	2	
Opis: manjkajoče ; # 21						
8.4.	2	40	načrtovanje	prevajanje	8	
Opis: napačen tip (int namesto float) – spremenljivka a # 5						
8.4.	3	20	kodiranje	prevajanje	1	
Opis: napačno ime spremenljivke (velike – male črke) – spremenljivka stLeta # 9						
10.4.	4					
Opis:						

PSP: Napake

Postopek izpolnjevanja Beležke napak:

1. Ob začetku razvoja programa izpolnimo podatke v glavi
2. Ko odkrijemo napako, najprej vnesemo samo številko napake
3. Vsako napako pišemo v svojo vrstico
4. Vnesemo datum, ko smo odkrili napako
5. Ko odpravimo napako, vnesemo najprej kategorijo napake
6. Vnesemo, v kateri fazi je bila napaka vnešena v program
7. Zabeležimo, v kateri fazi smo odpravili napako
8. Vnesemo tudi (oceno) časa (v min), ki smo ga potrebovali za odpravo napake
9. Stolpec [Št.pop.] zaenkrat pustimo prazen
10. Na kratko opišemo napako, da se je bomo kasneje lažje spomnili

PSP: Napake

Štetje napak:

- štejemo samo tiste napake, ki jih moramo odpraviti (ena napaka lahko povzroči veliko sporočil o napakah)
- napak ne odkrivamo samo ob prevajanju ali testiranju, temveč tudi pri načrtovanju in kodiranju
- ne glede na to se najprej osredotočimo na ti dve fazi, šele ko bomo bolj izkušeni, bomo posvetili pozornost načrtovanju in kodiranju
- če napako odkrijemo takoj, ko jo vnesemo, to ni napaka
- napake štejemo po zaključku faze (recimo po kodiranju procedure, izdelanem podatkovnem načrtu ipd.)
- verjetno bo največ napak vnešenih med načrtovanjem in kodiranjem
- verjetno bomo največ napak odkrili med prevajanjem in testiranjem

PSP: Napake

Pet razlogov za spremljanje (svojih) napak

1. Zaradi izboljšanja programerskega znanja
2. Zaradi zmanjšanja števila napak v programih
3. Zaradi prihranka časa (pri razvoju programov)
4. Zaradi manjših stroškov razvoja
5. Zaradi odgovornega opravljanja svojega dela

PSP: Napake

Napake beležimo tudi v Planu projekta

Verzijo 2 dopolnimo z naslednjimi podatki:

- skupno dosedanjo vrednostjo Min/LOC in LOC/Uro
- spremljanjem števila vnešenih napak
  - dejanskega števila vnešenih napak po fazah in skupaj
  - skupnega števila vnešenih napak doslej po fazah in skupaj
  - deleža števila vnešenih napak po fazah doslej
- spremljanjem števila odpravljenih napak
  - dejanskega števila odpravljenih napak po fazah in skupaj
  - skupnega števila odpravljenih napak doslej po fazah in skupaj
  - deleža števila odpravljenih napak po fazah doslej

PSP: Napake

PROJEKTNI PLAN (verzija 3) – 1/2

Izvajalec: _____ **Datum:** _____
Program: _____ **Zap.št:** _____
Vodja: _____ **Pr.jezik:** _____

SKUPAJ:	**Plan:**	**Dej.vrednost:**	**Skupaj doslej:**
Min/LOC:			
LOC/Uro:			

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):			
Največja velikost:			
Najmanjša velikost:			

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				
Načrtovanje				
Kodiranje				
Prevajanje				
Testiranje				
Analiza				
Skupaj:				
Največji čas:				
Najmanjši čas:				

PSP: Napake

PROJEKTNI PLAN (verzija 3) – 2/2

Izvajalec: _____ **Datum:** _____
Program: _____ **Zap.št:** _____
Vodja: _____ **Pr.jezik:** _____

VNEŠENE NAPAKE:	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje			
Načrtovanje			
Kodiranje			
Prevajanje			
Testiranje			
Analiza			
Skupaj:			

ODPRAVLJENE NAPAKE:	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje			
Načrtovanje			
Kodiranje			
Prevajanje			
Testiranje			
Analiza			
Skupaj:			

PSP: Napake

Z spremljanjem napak dopolnimo definiran proces razvoja PO

- opisan z **Navodilom za izvajanje PSP procesa**
- trenutno:
  - obsega 6 faz
  - predvideva uporabo obrazca Plan projekta
  - predvideva uporabo dveh beležk:
    - Beležke porabe časa
    - Beležke napak

Dopolnimo tudi pravila za izpolnjevanje Projektnega plana

- najlažje opišemo z **Navodilom za izpolnjevanje projektnega plana**
- verzija 3

Vnešene napake : odkrite napake : odpravljene napake!

PSP: Napake

Namen: postopek razvoja majhnih programov s spremljanjem kakovosti

Vhodni pogoj:

- opis problema
- Projektni plan
- zbrani podatki o dejanski porabi časa in velikosti preteklih projektov
- Beležka porabe časa
- Beležka napak

1. Planiranje

- pridobi opis značilnosti programa
- oceni velikost programa (pričakovano, največjo in najmanjšo)
- določi [Min/LOC]
- določi čas razvoja (pričakovan, največji in najmanjši)
- vnesi planirane vrednosti v Projektni plan
- vnesi čas planiranje v Beležko porabe časa

PSP: Napake

2. Načrtovanje

- načrtuj program
- opiši načrt programa v predvidenem formatu
- vnesi čas načrtovanja v Beležko porabe časa

3. Kodiranje

- Implementiraj načrt (s programom)
- uporabi standarden format pri kodiranju
- vnesi čas kodiranja v Beležko porabe časa

PSP: Napake

4. Prevajanje

- prevedi program
- zabeleži odkrite napake v Beležko napak
- odpravi vse odkrite napake
- vnesi čas prevajanja v Beležko porabe časa

5. Testiranje

- Testiraj program
- zabeleži odkrite napake v Beležko napak
- odpravi vse odkrite napake
- vnesi čas testiranja v Beležko porabe časa

PSP: Napake

6. Analizna faza

- do konca izpolni Projektni plan (dejanski čas razvoja, velikost, vsa razmerja in podatki o napakah)
- vnesi čas analize v Beležko porabe časa

Izhodni pogoj

- popolno testiran program (rezultat)
- popolno dokumentiran načrt
- celotna koda programa
- izpolnjen Projektni plan
- vnešeni ustrezni podatki v Beležki porabe časa
- vnešeni ustrezni podatki v Beležki napak

Navodilo za izpolnjevanje projektnega plana – 3 (1/5)

Namen: Obrazec vsebuje planirane in dejansko izmerjene podatke o projektu v čitljivi in uporabni obliki

Glava: Vnesite:

- ime razvijalca in datum začetka
- naziv in zaporedno številko programa
- ime vodje in programski jezik, v katerem bo napisan program

Min/LOC: Pred razvojem:

- vnesite planirano vrednost [Min/LOC]. Uporabite **doslej skupni Min/LOC** iz zadnjega projektnega plana ali beležke opravil.

Po razvoju:

- dejanski [Min/LOC] = [izmerjen čas] / [izmerjena velikost]
- **doslej skupni [Min/LOC] = [doslej skupni izmerjen čas] / [doslej skupna izmerjena velikost]**

Navodilo za izpolnjevanje projektnega plana – 3 (2/5)

LOC/Uro: Pred razvojem:

- planiran [LOC/Uro] = 60 / [Min/LOC]

Po razvoju:

- dejanski [LOC/Uro] = 60 / [Min/LOC]
- **doslej skupni [LOC/Uro] = 60 / doslej skupni [Min/LOC]**

Velikost (LOC): Pred razvojem:

- vnesite planirane vrednosti za skupno, največjo in najmanjšo velikost nove in spremenjene kode

Po razvoju:

- seštejte dejanske vrstice nove in spremenjene kode
- [doslej skupna velikost] = [dejanska velikost]+[doslej skupna velikost zadnjega projektnega plana]

Navodilo za izpolnjevanje projektnega plana - 3 (3/5)

Čas razvoja (min): Pred razvojem:

- določite planirane vrednosti časa razvoja po naslednjih formulah:
  [planiran skupni čas] = [planirana skupna velikost] * [Min/LOC]
  [planiran največji čas] = [planirana največja velikost] * [Min/LOC]
  [planiran najmanjši čas] = [planirana najmanjša velikost] * [Min/LOC]

Čas razvoja (min): Pred razvojem:

- poiščite [doslej %] vrednosti za vsako fazo posebej iz zadnjega projektnega plana
- izračunajte planirane čase razvoja na naslednji način:
  [planiran čas faze] = [planiran skupni čas] * [doslej % faze] / 100

Po razvoju:

- izmerite dejanski čas razvoja vsake faze in skupaj (s pomočjo Beležke porabe časa)

Navodilo za izpolnjevanje projektnega plana - 3 (4/5)

Čas razvoja (min): Po razvoju (nadaljevanje):

- [doslej skupni čas razvoja vsake faze in skupaj] =
  [dejanski čas vsake faze in skupaj]+
  [doslej skupen čas razvoja vsake faze in skupaj iz zad.projektnega plana]
- [doslej % skupni čas razvoja vsake faze] =
  100 * [doslej skupen čas razvoja vsake faze] /
  [doslej skupni čas razvoja skupaj]

Vnešene napake: Po razvoju:

- preštejte dejansko število vnešenih napak v vsaki fazi in skupaj (s pomočjo Beležke napak)
- [doslej skupno število vnešenih napak vsake faze in skupaj] =
  [dejansko število vnešenih napak vsake faze in skupaj]+
  [doslej skupno število vnešenih napak vsake faze in skupaj iz zad.proj.plan
- [doslej % skupni vnešenih napak vsake faze] =
  100 * [doslej skupno število vnešenih napak razvoja vsake faze] /
  [doslej skupno število vnešenih napak skupaj]

Navodilo za izpolnjevanje projektnega plana - 3 (5/5)

Odpravljene napake: Po razvoju:

- preštejte dejansko število odpravljenih napak v vsaki fazi in skupaj (s pomočjo Beležke napak)
- [doslej skupno število odpravljenih napak vsake faze in skupaj] = [dejansko število odpravljenih napak vsake faze in skupaj]+ [doslej skupno število odpravljenih napak vsake faze in skupaj iz zadnjega projektnega plana]
- [doslej % skupni odpravljenih napak vsake faze] = 100 * [doslej skupno število odpravljenih napak razvoja vsake faze] / [doslej skupno število odpravljenih napak skupaj]

PROJEKTNI PLAN (verzija 3) – 1/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	22 3. 2005
Program:	PROGRAM 7 (najboljši študi po letnikih UNI RI - PO)	**Zap.št:**	20
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**	**Skupaj doslej:**
Min/LOC:	8.26	5.43	6.65
LOC/Uro:	7.26	11.05	9.03

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	73	82	144
Največja velikost:	107		
Najmanjša velikost:	47		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	49	32	74	7.73
Načrtovanje	0	77	77	8.05
Kodiranje	370	240	554	57.89
Prevajanje	52	14	58	6.06
Testiranje	85	52	124	12.96
Analiza	65	30	70	7.31
Skupaj:	603	445	957	100.00
Največji čas:	884			
Najmanjši čas:	388			

PSP: Napake

PROJEKTNI PLAN (verzija 3) – 2/2

Izvajalec: ANDREJ NOVAK — Datum: 22 3. 2005
Program: PROGRAM 7 (najboljši študi po letnikih UNI RI - PO) — Zap.št: 20
Vodja: PRED. ROŽANC — Pr.jezik: JAVA

VNEŠENE NAPAKE:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje			0.00
Načrtovanje	2	2	25.00
Kodiranje	6	6	75.00
Prevajanje			0.00
Testiranje			0.00
Analiza			0.00
Skupaj:	8	8	100.00

ODPRAVLJENE NAPAKE:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje			0.00
Načrtovanje			0.00
Kodiranje			0.00
Prevajanje	6	6	75.00
Testiranje	2	2	25.00
Analiza			0.00
Skupaj:	8	8	100.00

PSP: Odkrivanje napak

Z beleženjem napak spoznamo (razumemo) napake

PSP omogoča učinkovito odkrivanje in odpravljanje napak

Dve ugotovitvi:

a) Praktično nemogoče je odkriti čisto vse napake

b) Napake, ki so vnešene zgodaj v razvojnem procesu ali so dolgo prisotne, imajo hujše posledice

Dva cilja:

a) Z ustreznimi pristopi poskušajmo odkriti kar največ napak

b) Pomembno je, da napake odkrijemo čim bolj zgodaj v razvojnem procesu

PSP: Odkrivanje napak

Koraki za odkrivanje napak:

1) Prepoznamo (tipične) simptome napake ali njene posledice
2) Na podlagi simptomov sklepamo, za kakšno napako gre
3) Konkretno preverimo, kje v programu se napaka nahaja
4) Odločimo se, kako bomo odpravili napako
5) Popravimo program (odpravimo napako)
6) Preverimo, če je popravek odpravil težavo (ni več simptomov)

Trije običajni načini pri odkrivanju napak:

a) pri prevajanju
b) pri testiranju
c) napake odkriva uporabnik

PSP: Odkrivanje napak

a) pri prevajanju

- prvi pristop
- namen prevajalnika ni odkrivanje napak
- delovanje prevajalnika: odkrijemo sintaktične napake
- odkrijemo samo del napak (cca. 10% sintaktičnih napak uide)
- logičnih (najtežjih) napak običajno ne odkrijemo
- odkrivamo samo simptome napak – vzroki ostajajo skriti
- problem zavajajočih sporočil ob napakah
- nujen del odkrivanja napak, ki običajno ne zadošča

PSP: Odkrivanje napak

b) pri testiranju

- namen: odkrivanje napak
- dva ključna dela: testni podatki in pogoji testiranja (testni scenariji)
- odvisnost od osebe, ki izvaja testiranje: namen, temejitost, natančnost
- veliko vrst testiranj

Težave:

- odkrivamo simptome, ne vzrokov
- vsak test preverja eno stvar
- problem "kritičnih poti", mejnih vrednosti
- testiranje je časovno potratno – smiselen obseg
- za vse razen najenostavnejših programov je zares popolno testiranje nemogoče

PSP: Odkrivanje napak

c) napake odkriva uporabnik

- zelo pogosto
- ne moremo odpraviti vseh napak, v vsakem primeru se zgodi
- strategije različic programa za testiranje
- zelo visoki stroški

Zelo priporočljiv pristop za posameznika: (ponovni) pregled kode

- ko je koda (ali načrt) dokončana - običajno pred naslednjo fazo (recimo prevajanjem),
- Če se le da natisnemo kodo na papir ("listing")
- jo natančno vrstico za vrstico pregledamo

Čeprav (navidez brez potrebe) porabimo precej časa, se ta pristop izkaže za zelo hiter in učinkovit način odkrivanja napak

PSP: Odkrivanje napak

Prednosti pregledovanja kode:

- pregledujemo rešitev (problem) in ne iščemo simptomov napak
- ob zaključku kodiranja je ideja rešitve še sveža
- izpis na papirju omogoča lažji pregled, skoke, beležke opomb
- učinkoviteje kot testiranje 3 – 5 krat: 2-5 napak/uro proti 6-10 napak/uro
- privarčujemo čas, ki ga pri testiranju porabimo za iskanje napak na podlagi odkritih simptomov

Slabosti:

- pregledovanje kode je časovno potratno
- pravilna izvedba je zahtevna in zahteva veščino
- običajno bomo odkrili 70-80% napak
- potrebujemo najmanj 30 minut za 100 LOC
- ustrezen pristop pomaga (PSP)

PSP: Odkrivanje napak

Zakaj je treba napake odkrivati zgodaj?

- napake se v izdelek vgrajujejo postopoma med procesom izdelave
- če ima izdelek vgrajenih veliko napak, zelo težko ob koncu razvoja zagotovimo kakovost

Primer: raje kupimo avtomobil, ki ima na testiranju manj napak

Cena odkrivanja in odpravljanja napak

Tipičen postopek testiranja:

- testiranje posemeznega modula (unit testing)
- združitveno testiranje ob sestavljanju modulov (integration testing)
- sistemsko testiranje, ko več komponet sestavimo v sistem (system testing)

Če upoštevamo še ceno napake po izročitvi izdelka uporabniku velja, da je odprava napake na vsaki višji stopnji 10 krat dražja !

PSP: Odkrivanje napak

Na to vpliva:

- izdelek postaja vedno bolj obsežen in zahteven
- tudi testiranja postajajo vedno zahtevnejša in dražja
- (po eni strani) kasneje odkrivamo tipično hujše napake (načrtovanja...)
- (po drugi strani) pa se spremeni nivo preverjanja, zato težko odkrivamo enostavne napake, ki so ostale (recimo sintaksne napake ob sistemskem testiranju)

Poleg tega velja, da je učinkovitost odkrivanja napak na vsakem višjem nivoju manjša:

- prevajanje: cca 90% (sintaktičnih) napak
- testiranje modulov: cca 50% obstoječih napak
- združitveno testiranje: cca 40 % obstoječih napak
- sistemsko testiranje: cca 30 % obstoječih napak

PSP: Odkrivanje napak

Izvajanje pregledov kode v našem primeru:

- izkušnje: 5 do 10 zmanjšamo število napak pri prevajanju in testiranju
- nujno: na podlagi zbranih podatkov o napakah moramo razbrati, kakšne napake delamo
- postopek odkrivanja prilagodimo posamezniku (slabo tipkanje)
- težimo k temu, da napake odkrijemo in odpravimo čimprej, zato:
  - pregled kode izvedemo pred prevajanjem
  - vedno preverjamo na listu natisnjen listing
  - vsako odkrito napako zabeležimo v beležko napak
  - pregled kode prilagodimo napakam, ki smo jih nazadnje pogosto odkrivali

Posebna navodila za preverjanje kode

Navodilo za izvajanje preverjanja kode (1/4)

Vhodni pogoj:
- opis problema (seznam zahtev)
- načrt programa
- programska koda
- Standard za kodiranje
- Beležka porabe časa, Beležka napak

1. Postopek pregleda
- dokončaj programsko kodo
- pred prevajanjem ali testiranjem, natisni kodo na papir
- preveri programsko kodo vrstico za vrstico
- ob preverjanju vrstic sproti odpravi vsako napako, ki jo odkriješ
- sproti vnašaj porabljen čas v Beležko porabe časa

Navodilo za izvajanje preverjanja kode (2/4)

2. Odprava napak
- odpravi vse odkrite napake
- preveri popravke, da so napake zares odpravljene
- vnesi vse napake v Beležko napak

3. Globalni pregled
- preveri, če načrt programa vsebuje vse funkcije, ki so zapisane v zahtevah
- preveri, če programska koda vsebuje vse funkcije iz načrta

4. Pregled programske logike
- preveri, če je logika načrta programa pravilna
- preveri, če programska koda pravilno implementira logiko načrta

Navodilo za izvajanje preverjanja kode (3/4)

5. Pregled imen in tipov

- preveri, če so imena in tipi pravilno deklarirani in uporabljeni
- preveri ustrezne deklaracije naslednjih tipov podatkov: int, long, short, byte, float, double, char in String

6. Pregled spremenljivk

- preveri, če so vse spremenljivke inicializirane na ustrezne začetne vrednosti
- preveri, če lahko spremenljivke dobijo vrednosti, ki so izven obsega tipa

Navodilo za izvajanje preverjanja kode (4/4)

7. Pregled sintakse

- preglej, če je vsaka vrstica dosledno upošteva sintaksna pravila programskega jezika

Izhodni pogoj

- celotna koda programa, ki vsebuje vse popravke
- vnešeni ustrezni podatki v Beležki porabe časa
- vnešeni ustrezni podatki v Beležki napak

PSP: Odkrivanje napak

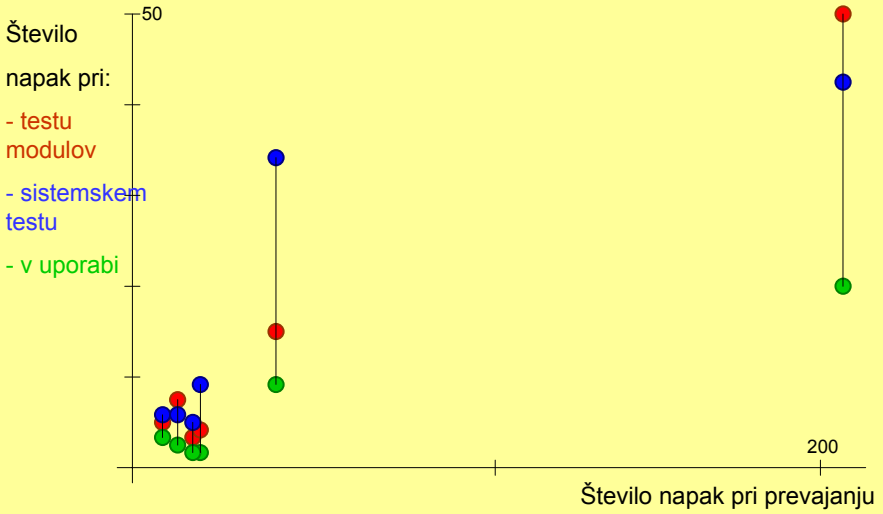
Zakaj je primerno pregledovati kodo pred prevajanjem?

- za to porabimo približno enako časa kot po prevajanju
- pregled bo privarčeval veliko časa za prevajanje: brez pregledov porabimo za to 12 – 15%, z pregledi pa le cca 3%
- ko se program uspešno prevede, nismo več tako motivirani za pregledovanje in pregledi so zato bolj površni
- učinkovitost pregledo pred in po prevajanju je približno enaka
- če je veliko napak pri prevajanju, jih bo tudi pri testiranju

Primer:

- sistem z šestimi programskimi komponentami
- vse med 600 in 2500 LOC
- vsi razvijalci razen enega obvladajo PSP in pregledujejo kodo
- graf kaže razmerje med številom napak pri prevajanju in številom odkritih napak pri testiranju modulov, sistemskem testiranju in uporabi

PSP: Odkrivanje napak



PSP: Odkrivanje napak

Pregled kode je dodatna faza:

- po kodiranju in pred prevajanjem
- **Navodilo za izvajanje PSP** dopolnimo s:

3. Kodiranje

...

4. Pregled kode

- temeljito preglej programsko kodo
- upoštevaj Navodilo za pregledovanje kode
- odpravi in zabeleži vsako odkrito napako (v Beležko napak)
- vnesi čas pregledovanja kode v Beležko porabe časa

5. Prevajanje

...

PSP: Odkrivanje napak

Tudi plan projekta dopolnimo – verzija 4:

- dodamo novo fazo "Pregled kode" za:
  - porabo časa
  - število vnešenih napak
  - število odpravljenih napak
- Navodilo za izpolnjevanje plana projekta se ne spremeni
- primer ...

TSP: preglede kode nadgradimo s pregledi med kolegi (peer reviews):

- kolegi odkrijejo napake, ki jih sami ne vidimo (2 razloga)
- na ta način odkrivamo tudi napake, ki jih posameznik ne bi mogel
- nujno dobro sodelovanje v skupini
- še vedno lahko ohranimo preglede kode

PSP: Odkrivanje napak

PROJEKTNI PLAN (verzija 4) – 1/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	29. 3. 2005
Program:	PROGRAM 8	**Zap.št:**	24
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**	**Skupaj doslej:**
Min/LOC:	6.65	6.64	6.64
LOC/Uro:	9.03	9.04	9.04

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	112	102	246
Največja velikost:	167		
Najmanjša velikost:	88		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	58	42	116	7.10
Načrtovanje	60	57	134	8.20
Kodiranje	431	440	994	60.83
Pregled kode	0	45	45	2.75
Prevajanje	45	10	68	4.16
Testiranje	97	48	172	10.53
Analiza	54	35	105	6.43
Skupaj:	745	677	1634	100.00
Največji čas:	1111			
Najmanjši čas:	585			

PSP: Odkrivanje napak

PROJEKTNI PLAN (verzija 4) – 2/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	29 3. 2005
Program:	PROGRAM 8	**Zap.št:**	24
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

VNEŠENE NAPAKE:	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje			0.00
Načrtovanje	1	3	21.43
Kodiranje	5	11	78.57
Pregled kode			0.00
Prevajanje			0.00
Testiranje			0.00
Analiza			0.00
Skupaj:	6	14	100.00

ODPRAVLJENE NAPAKE:	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje			0.00
Načrtovanje			0.00
Kodiranje			0.00
Pregled kode	3	3	21.43
Prevajanje	2	8	57.14
Testiranje	1	3	21.43
Analiza			0.00
Skupaj:	6	14	100.00

PSP: Pregled programske kode

Pri pregledu programske kode pomaga opomnik (checklist):

- opomnik ima značilnosti standarda in beležke
- pomaga, da natančno izvedemo vse predvidene korake
- namenjen za določeno vrsto programske kode
- omogoča prilagajanje posamezniku
- vsebuje osebne izkušnje: beležimo dejansko število odkritih napak
- opomnik moramo nadgrajevati

Dva primera:

- opomnik za programsko kodo, napisano v prog. jeziku C++
- opomnik za programsko kodo, napisano v prog. jeziku ADA

PSP predvideva, da uporabljamo opomnik pri pregledovanju kode

PSP: Pregled programske kode

Program Name and #:

Purpose	To guide you in conducting an effective code review.	#	#	#	#	To Date	To Date %
General	As you complete each review step, note the number of defects of that type found in the box to the right. If none, put a check in the box at the right. Complete the checklist for one program, class, object, or method before you start to review the next.						
Complete	Verify that all the functions in the design are coded.						
Includes	Verify that `includes` are complete.						
Initialization	Check variable and parameter initialization: • at program initiation • at start of every loop • at function/procedure entry						
Calls	Check function call formats: • Pointers • Parameters • Use of '&'						
Names	Check name spelling and use: • Is it consistent? • Is it within declared scope? • Do all structures/classes use "." reference?						
Strings	Check that all strings are: • identified by pointers • terminated in NULL						
Pointers	Check that pointers are: • initialized NULL • only deleted after new • always deleted after use if new						
Output format	Check the output format: • Is line stepping proper? • Is spacing proper?						
{} Pairs	Ensure that the {} are proper and matched.						
Logic operators	Verify the proper use of ==, =, ||, etc. Check every logic function for proper ()						
Line-by-line check	Check every line of code: • Instruction syntax • Proper punctuation						
Standards	Ensure that the code conforms to the coding standards.						
File open and close	Verify that all files are: • properly declared • opened • closed						
Overall	Do an overall scan of the program to check system issues and unexpected problems.						

PSP: Pregled programske kode

Program Name and #:

Purpose	To guide you in conducting an effective code review.	#	#	#	#	To Date	To Date %
General	As you complete each review step, note the number of defects of that type found in the box to the right. If none, put a check in the box at the right. Complete the checklist for one program, class, object, or method before you start to review the next.						
Complete	Verify that all the functions in the design are coded.						
Includes	Verify that the `with` statements are complete.						
Initialization	Check variable and parameter initialization: • at program initiation • at start of every loop • at procedure entry						
Calls	Check procedure call formats: • Punctuation • Parameters						
Names	Check name spelling and use: • Is it consistent? • Is it within declared scope? • Do all structures/packages use '.' reference?						
Strings	Check that all strings make proper use of slices.						
Pointers	Check that pointers are: • only deleted after new • always deleted after use if new						
Output format	Check the output format: • Is line stepping proper? • Is spacing proper?						
() Pairs	Ensure that the () are proper and matched.						
Logic operators	Verify the proper use of all logic operators. Check every logic function for proper ()						
Line-by-line check	Check every line of code: • Instruction syntax • Proper punctuation						
Standards	Ensure that the code conforms to the coding standards.						
File open and close	Verify that all files are: • properly declared • opened • closed						
Overall	Do an overall scan of the program to check for system issues and unexpected problems.						
Totals							

PSP: Pregled programske kode

Uporaba opomnika:

1. Za vsako točko v opomniku (tip napake) preverimo celotno kodo ter zabeležimo in odpravimo vse odkrite napake.
2. Vsako odkrito napako zabeležimo s črtico v opomniku v stolpcu #.
3. Če ob pregledu za točko ni bilo odkritih napak, vnesemo v stoplec # X.
4. Če je program sestavljen iz več modulov (procedur, funkcij), je smiselno preveriti vsako posebej po vseh točkah. V tem primeru vnesemo število napak v več stolpcev #.
5. Preverimo programsko kodo kot celoto. Upoštevamo tudi druge napake, ki mogoče niso v opomniku.
6. Na koncu seštejemo črtice ob vsaki točki.

Na podlagi zapisanega dopolnimo **Navodilo za izvedbo preverjanja kode**

PSP: Pregled programske kode

Navodilo za izvajanje preverjanja kode (1/4)

Vhodni pogoj:

- opis problema (seznam zahtev)
- ...
- Beležka porabe časa, Beležka napak
- izvod Opomnika za pregled kode

Splošno

- uporabljaj opomnik za pregled kode
- preglede izvajaj po navodilih opomnika, točko za točko
- ob zaključku pregleda določi stolpca **[Skupaj doslej]** in **[Doslej%]** ter vrstico **[Skupaj]**

1. Postopek pregleda

- ...

PSP: Pregled programske kode

Prilagajanje opomnika za svoje potrebe (1/2):

1. Izdelajmo tabelo s številom napak in kategorijami napak (od 10 do 100), ki so bile vnešene in odpravljene v vsaki fazi (za več zadnjih programov)
2. Uredimo kategorije napak padajoče glede na število odkritih napak pri prevajanju in testiranju.
3. Za kategorije, kjer je bilo največ napak, preverimo beležko napak in določimo ključne napake.
4. Za ključne napake opredelimo načine za njihovo odkrivanje pri pregledovanju kode.
5. V opomnik vnesemo točke, ki predstavljajo ustrezne načine pregleda za odkrivanje ključnih napak.
6. S spremenjenim opomnikom ponovno pregledamo že analizirane programe.

PSP: Pregled programske kode

Prilagajanje opomnika za svoje potrebe (2/2):

7. Če je opomnik našel vse ključne napake, ga v bodoče uporabljamo kot stalni opomnik.
8. Če opomnik ni omogočil najti določenih napak, ponovimo začetne korake in bolje opredelimo ključne vrste napak ter njihovo odkrivanje.
9. Pri popravljanju opomnika je smiselno združevati podobne vrste napak. Bodimo pozorni na nejasnosti in podvajanja.
10. Ob vsakem pregledu kode na hitro preverimo opomnik in ga po potrebi dopolnimo.
11. Razmislimo, kako bi v bodoče preprečili nastanek ključnih napak (z boljšim planiranjem, načrtovanjem, ...) in to uvedemo v prakso.

Primer prilagojenega opomnika po pregledu kode

PSP: Pregled programske kode

Kategorija	Naziv	Opis
10	Dokumentacija	komentarji, sporočila, opis delov programa
20	Sintaksa	tipkarske napake, ločila, format izpisa podatkov
30	Izgradnja	obvladovanje zahtev, moduli, knjižnice, verzije
40	Prirejanje	deklaracije, podvojena imena, veljavnost pravil, omejitve
50	Vmesnik	klici podprogramov, V/I, format podatkov
60	Preverjanje	obvladovanje napak, sporočila, pomankljivo preverjanje
70	Podatki	struktura, vsebina
80	Funkcije	logika, kazalci, zanke, rekurzija, izračuni, fun. napake
90	Sistem	konfiguracija, časovna razporeditev, spomin
100	Okolje	načrt, prevajanje, testiranje, podpora sistemu

PSP: Pregled programske kode

Kat	Vnešene napake			Odstranjene napake			Zgrešene
	Načrt	Koda	Ostalo	Pregled	Prevajanje	Testiranje	Pri pregledu
10							
20		8		4	4		4
40	2	3		1	4		4
50		2			1	1	2
80	2	3			1	4	5
Sk.	4	16		5	10	5	15
Program							
10	2	6			6	2	8
11	1	5		3	2	1	3
12	1	5		2	2	2	4

PSP: Pregled programske kode

Kat	Vnešene napake			Odstranjene napake			Zgrešene
	Načrt	Koda	Ostalo	Pregled	Prevajanje	Testiranje	Pri pregledu
80	2	3			1	4	5
20		8		4	4		4
40	2	3		1	4		4
50		2			1	1	2
10							
30							
60							
...							
100							
Sk.	4	16		5	10	5	15

PSP: Pregled programske kode

Program Name and #:

Purpose	To guide you in conducting an effective code review.	#	#	#	#	To Date	To Date %
General	As you complete each review step, note the number of defects of that type found in the box to the right. If none, put a check in the box at the right. Complete the checklist for one program, class, object, or method before you start to review the next.						
Complete	Verify that all the functions in the design are coded.	x					
Includes	Verify that the #include statements are complete.	x					
Initialization	Check variable and parameter initialization: • at program initiation • at start of every loop • at procedure entry	x					
Calls	Check procedure call formats: • Pointers • Parameters	x					
Names	Check name spelling and use: • Is it consistent? • Is it within declared scope? • Do all structures/packages use '.' reference?	1				2	40
Strings	Check that all strings are: • identified by pointers • terminated in NULL	x					
Pointers	Check that pointers are: • initialized NULL • only deleted after new • always deleted after use if new	x					
Output format	Check the output format: • Is line stepping proper? • Is spacing proper?	x					
() Pairs	Ensure that the () are proper and matched.	x					
Logic operators	Verify the proper use of all logic operators. Check every logic function for proper ()	x					
Line-by-line check	Check every line of code: • Instruction syntax • Semicolons are properly used • Check that semicolons are not typed as colons • Other punctuation	1				3	60
Standards	Ensure that the code conforms to the coding standards.	x					
File open and close	Verify that all files are: • properly declared • opened • closed	x					
Overall	Do an overall scan of the program to check system issues and unexpected problems.	x					

PSP: Pregled programske kode

Standard za kodiranje:

- obsega definiran nabor pravil, ki se jih držimo pri kodiranju
- dodatna pravila za večjo jasnost/razumljivost programske kode
- se razlikuje od opomnika, ki tudi temelji na pravilih programskega jezika
- dopolnjuje opomnik oziroma lajša njegovo uporabo
- deluje preventivno zoper vnašanje napak
- vsebuje pomembne razlike med programskimi jeziki
- obstajajo že znani primeri, vendar standard lahko prilagodimo za svojo uporabo

Primer: standard za kodiranje v programskem jeziku C++

PSP: Pregled programske kode

Purpose	To guide the development of C++ programs
Program headers	All programs begin with a descriptive header.

Header format	`/*************************************************/`
	`/* Program Assignment: the program number */`
	`/* Name: your name */`
	`/* Date: the date program */`
	`/* development started */`
	`/* Description: a short description of */`
	`/* the program */`
	`/* function */`
	`/*************************************************/`

Listing contents	Provide a summary of the listing contents.

Contents example	`/*************************************************/`
	`/* Listing Contents: */`
	`/* Reuse instructions */`
	`/* Includes */`
	`/* Class declarations: */`
	`/* CData */`
	`/* ASet */`
	`/* Source code in c:\classes\CData.cpp: */`
	`/* CData */`
	`/* CData() */`
	`/* Empty() */`
	`/*************************************************/`

Reuse instructions	Describe how the program is used. Provide the declaration format, parameter values and types, and parameter limits. Provide warnings of illegal values, overflow conditions, or other conditions that could potentially result in improper operation.

Example	`/*************************************************/`
	`/* Reuse Instructions */`
	`/* int PrintLine(char *line_of_character) */`
	`/* Purpose: to print string, */`
	`/* 'line_of_character', on one print line */`
	`/* Limitations: the maximum line length is */`
	`/* LINK_LENGTH */`
	`/* Return: 0 if printer not ready to print, */`
	`/* else 1 */`
	`/*************************************************/`

Identifiers	Use descriptive names for all variables, function names, constants, and other identifiers. Avoid abbreviations or single letter variables.
Identifier example	`int number_of_students; /* This is GOOD */` `float x4, j, ftave; /* These are BAD */`

PSP: Pregled programske kode

Comments	Sufficiently document the code so the reader can understand its operation. Comments should explain both the purpose and behavior of the code. Comment variable declarations to indicate their purpose.
Good comment	`if(record_count > limit) /* have all the */` `/* records been processed? */`
Bad comment	`if(record_count > limit) /* check if record_ */` `/* count is greater than limit */`
Major sections	Major program sections should be preceded by a block comment that describes the processing that is done in the next section.

Example	`/*************************************************/`
	`/* This program section will examine the */`
	`/* contents of the array "grades" */`
	`/* and will calculate the average grade */`
	`/* for the class */`
	`/*************************************************/`

Blank space	Write programs with sufficient spacing so they are easy to read. Separate every program construct with at least one space.
Indenting	Indent every level of bracket from the previous one. Open and closing brackets should be on lines by themselves and aligned with each other.

Indenting example	`while (miss_distance > threshold)`
	`{`
	`success_code=move_robot (target_location);`
	`if (success_code==MOVE_FAILED)`
	`{`
	`printf("The robot move has failed.\n");`
	`}`
	`}`

Capitalization	All defines are capitalized. All other identifiers and reserved words are lowercase. Messages being output to the user can be mixed-case so as to make a clean user presentation.

Capitalization example	`#define DEFAULT_NUMBER_OF_STUDENTS 15`
	
	`int class_size=DEFAULT_NUMBER_OF_STUDENTS;`

PSP: Planiranje napak

Namen:

- predstavitev načina za analiziranje podatkov o napakah
- izboljšuje natančnost planiranja in kakovost izdelkov
- opredeljuje preventivne ukrepe za zmanjševanje napak

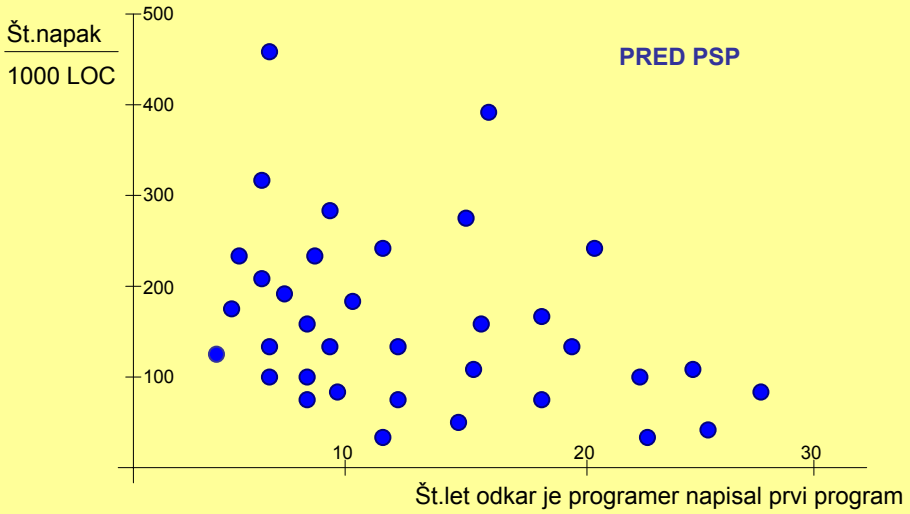
Izkušen programer vnese med 50 in 250 napak na 1000 LOC

Primer:

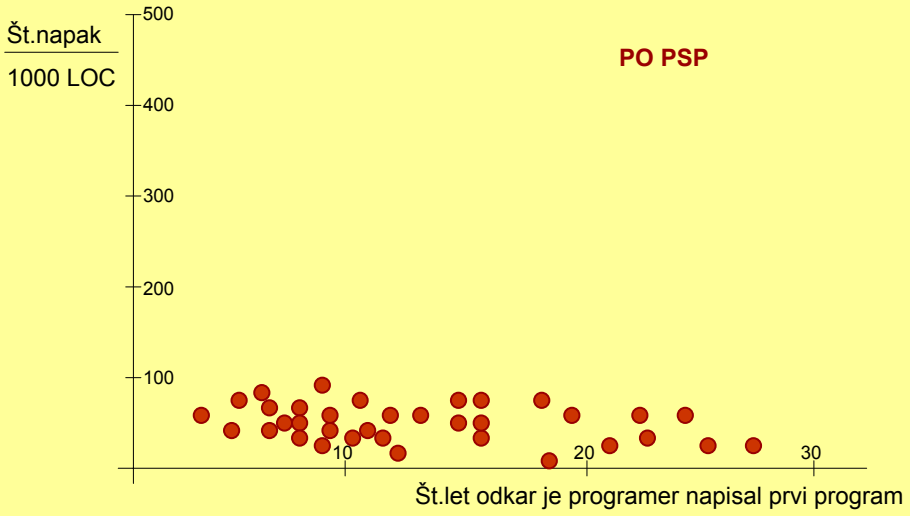
- 38 programerjev pred in po PSP usposabljanju
- prikaz odvisnosti med številom napak in leti, od kar je programer napisal svoj prvi program
- A: majhen programski projekt pred PSP
- B: 10 majhnih programov z PSP

Dejstvo: tudi izkušeni programerji z pravimi prijemi vnašajo veliko napak

PSP: Planiranje napak



PSP: Planiranje napak



PSP: Planiranje napak

Uporaba podatkov o napakah

- namen spremljanja napak je ocenjevanje števila napak, ki jih bomo vnesli v nov program
- pomembno za odločanje o dodatnih pregledih kode in testiranju
- omogoča določanje in odločanje o kakovosti izdelka
- nujnost ustreznega izvajanja (osebna zaveza)

Gostota napak:

- osnovna mera za količino napak
- merimo število napak na 1000 LOC (= 1 KLOC)
- uporabimo :
  - dejanska velikost programa (število novih in spremenjenih vrstic)
  - skupno število vnešenih (odkritih) napak v programu

PSP: Planiranje napak

Primer:

- izmerjena velikost programa: 96 LOC
- skupno število vnešenih napak: 14

[Gostota napak] = 1000 *
[skupno število vnešenih napak] /
[dejanska velikost programa]

[Gostota napak] = 1000 * 14 / 96 = 145.83 napak/KLOC

Planiranje napak je zahtevno, ker:

- se naše sposobnosti programiranja izboljšujejo
- proces razvoja še ni stabilen
- je čas odpravljanja napak le približno odvisen števila napak

PSP: Planiranje napak

Planiranje s pomočjo Plana projekta:

- kot doslej planiramo velikost programa, čas razvoja skupaj na podlagi znanih razmerij in podatkov preteklih projektov
- določimo predvideno [Število napak/KLOC] glede na doslej skupno število napak in doslej skupno velikost vseh programov
- določimo pričakovano število napak v programu kot
  1000 * [Število napak/KLOC] / planirana velikost programa
- v skladu z [% doslej napak] razporedimo napake po posameznih fazah

Plan projekta:

- verzija 5
- dodana vrstica za planiranje, izmerjene in doslej skupne vrednosti [Števila napak/KLOC]
- dodano planiranje števila napak po fazah in skupaj

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 1/2

Izvajalec: _____ **Datum:** _____
Program: _____ **Zap.št:** _____
Vodja: _____ **Pr.jezik:** _____

SKUPAJ:	**Plan:**	**Dej.vrednost:**	**Skupaj doslej:**
Min/LOC:			
LOC/Uro:			
Št.napak/KLOC:			

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):			
Največja velikost:			
Najmanjša velikost:			

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				
Načrtovanje				
Kodiranje				
Pregled kode				
Prevajanje				
Testiranje				
Analiza				
Skupaj:				
Največji čas:				
Najmanjši čas:				

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 2/2

Izvajalec: _____ **Datum:** _____
Program: _____ **Zap.št:** _____
Vodja: _____ **Pr.jezik:** _____

VNEŠENE NAPAKE:	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				
Načrtovanje				
Kodiranje				
Pregled kode				
Prevajanje				
Testiranje				
Analiza				
Skupaj:				

ODPRAVLJENE NAPAKE:	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				
Načrtovanje				
Kodiranje				
Pregled kode				
Prevajanje				
Testiranje				
Analiza				
Skupaj:				

PSP: Planiranje napak

Ustrezno dopolnimo Navodilo za izvajanje procesa:

- 7 faz
- formalno sprememba le v fazi planiranja

Dopolnimo tudi Navodilo za izpolnjevanje Plana projekta:

- način določanja vseh novih vrednosti
- opredelitev planiranja in vnosa dejanskih vrednosti

PSP: Planiranje napak

Namen: postopek razvoja majhnih programov s spremljanjem kakovosti

Vhodni pogoj:

- opis problema
- Projektni plan
- zbrani podatki o dejanski porabi časa in velikosti preteklih projektov
- Beležka porabe časa
- Beležka napak

1. Planiranje

- pridobi opis značilnosti programa
- oceni velikost programa (pričakovano, največjo in najmanjšo)
- določi [Min/LOC], [LOC/Uro] in [Št.napak/KLOC]
- določi čas razvoja (pričakovan, največji in najmanjši)

PSP: Planiranje napak

1. Planiranje (nadaljevanje)
- določi pričakovano število napak (po fazah in skupaj)
- vnesi planirane vrednosti v Projektni plan
- vnesi čas planiranje v Beležko porabe časa

2. Načrtovanje
- načrtuj program
- opiši načrt programa v predvidenem formatu
- vnesi čas načrtovanja v Beležko porabe časa

3. Kodiranje
- Implementiraj načrt (s programom)
- uporabi standarden format pri kodiranju
- vnesi čas kodiranja v Beležko porabe časa

PSP: Planiranje napak

4. Pregled kode
- temeljito preglej programsko kodo
- upoštevaj Navodilo za pregledovanje kode
- odpravi in zabeleži vsako odkrito napako (v Beležko napak)
- vnesi čas pregledovanja kode v Beležko porabe časa

5. Prevajanje
- prevedi program
- zabeleži odkrite napake v Beležko napak
- odpravi vse odkrite napake
- vnesi čas prevajanja v Beležko porabe časa

6. Testiranje
- testiraj program
- zabeleži odkrite napake v Beležko napak

6. Testiranje (nadaljevanje)

- odpravi vse odkrite napake
- vnesi čas testiranja v Beležko porabe časa

7. Analizna faza

- do konca izpolni Projektni plan (dejanski čas razvoja, velikost, vsa razmerja in podatki o napakah)
- vnesi čas analize v Beležko porabe časa

Izhodni pogoj

- popolno testiran program (rezultat)
- popolno dokumentiran načrt
- celotna koda programa
- izpolnjen Projektni plan
- vnešeni ustrezni podatki v Beležki porabe časa
- vnešeni ustrezni podatki v Beležki napak

Navodilo za izpolnjevanje projektnega plana - 5 (1/7)

Namen: Obrazec vsebuje planirane in dejansko izmerjene podatke o projektu v čitljivi in uporabni obliki

Glava: Vnesite:

- ime razvijalca in datum začetka
- naziv in zaporedno številko programa
- ime vodje in programski jezik, v katerem bo napisan program

Min/LOC: Pred razvojem:

- vnesite planirano vrednost [Min/LOC]. Uporabite doslej skupni Min/LOC iz zadnjega projektnega plana ali beležke opravil.

Po razvoju:

- dejanski [Min/LOC] = [izmerjen čas] / [izmerjena velikost]
- doslej skupni [Min/LOC] = [doslej skupni izmerjen čas] / [doslej skupna izmerjena velikost]

Navodilo za izpolnjevanje projektnega plana – 5 (2/7)

LOC/Uro: Pred razvojem:
- planiran [LOC/Uro] = 60 / [Min/LOC]

Po razvoju:
- dejanski [LOC/Uro] = 60 / [Min/LOC]
- doslej skupni [LOC/Uro] = 60 / doslej skupni [Min/LOC]

Št.napak/KLOC: Pred razvojem:
- planirano [Št.napak/KLOC] = [Doslej skupaj št.napak/KLOC] iz zadnjega projektnega plana

Po razvoju:
- dejansko [Št.napak/KLOC] = 1000 * [dejansko število napak skupaj] / [dejanska velikost]
- [Doslej skupaj št.napak/KLOC] = 1000 * [doslej skupaj število napak] / [doslej skupna velikost]

Navodilo za izpolnjevanje projektnega plana – 5 (3/7)

Velikost (LOC): Pred razvojem:
- vnesite planirane vrednosti za skupno, največjo in najmanjšo velikost nove in spremenjene kode

Po razvoju:
- seštejte dejanske vrstice nove in spremenjene kode
- [doslej skupna velikost] = [dejanska velikost]+[doslej skupna velikost zadnjega] projektnega plana

Čas razvoja (min): Pred razvojem:
- določite planirane vrednosti časa razvoja po naslednjih formulah:
  [planiran skupni čas] = [planirana skupna velikost] * [Min/LOC]
  [planiran največji čas] = [planirana največja velikost] * [Min/LOC]
  [planiran najmanjši čas] = [planirana najmanjša velikost] * [Min/LOC]

Navodilo za izpolnjevanje projektnega plana – 5 (4/7)

Čas razvoja (min): Pred razvojem (nadaljevanje):

- poiščite [doslej %] vrednosti za vsako fazo posebej iz zadnjega projektnega plana
- izračunajte planirane čase razvoja na naslednji način:
  [planiran čas faze] = [planiran skupni čas] * [doslej % faze] / 100

Po razvoju:

- izmerite dejanski čas razvoja vsake faze in skupaj
  (s pomočjo Beležke porabe časa)
- [doslej skupni čas razvoja vsake faze in skupaj] =
  [dejanski čas vsake faze in skupaj] +
  [doslej skupen čas razvoja vsake faze in skupaj iz zad.projektnega plana]
- [doslej % skupni čas razvoja vsake faze] =
  100 * [doslej skupen čas razvoja vsake faze] /
  [doslej skupni čas razvoja skupaj]

Navodilo za izpolnjevanje projektnega plana – 5 (5/7)

Vnešene napake: Pred razvojem:

- določite planirano število vnešenih napak (po fazah in skupaj)
- planirano [število vnešenih napak skupaj] = planirano [Št.napak/KLOC] * planirana [skupno število vrstic nove in spremenjene kode] / 1000
- poiščite [doslej %] števila vnešenih napak za vsako fazo posebej iz zadnjega projektnega plana
- planirano [število vnešenih napak faze] = planirano [skupno število vnešenih napak] * [doslej % vnešenih napak faze] / 100

Po razvoju:

- preštejte dejansko število vnešenih napak v vsaki fazi in skupaj
  (s pomočjo Beležke napak)
- [doslej skupno število vnešenih napak vsake faze in skupaj] =
  [dejansko število vnešenih napak vsake faze in skupaj]+
  [doslej skupno število vnešenih napak vsake faze in skupaj iz zad.pr.plana]

Navodilo za izpolnjevanje projektnega plana – 5 (6/7)

Vnešene napake: Po razvoju (nadaljevanje):

- [doslej % skupni vnešenih napak vsake faze] = 100 * [doslej skupno število vnešenih napak razvoja vsake faze] / [doslej skupno število vnešenih napak skupaj]

Odpravljene napake: Pred razvojem:

- določite planirano število odpravljenih napak (po fazah in skupaj)
- planirano [število odpravljenih napak skupaj] = planirano [število vnešenih napak skupaj]
- poiščite [doslej %] števila odpravljenih napak za vsako fazo posebej iz zadnjega projektnega plana
- planirano [število odpravljenih napak faze] = planirano [skupno število odpravljenih napak] * [doslej % odpravljenih napak faze] / 100

Navodilo za izpolnjevanje projektnega plana – 5 (7/7)

Odpravljene napake: Po razvoju:

- preštejte dejansko število odpravljenih napak v vsaki fazi in skupaj (s pomočjo Beležke napak)
- [doslej skupno število odpravljenih napak vsake faze in skupaj] = [dejansko število odpravljenih napak vsake faze in skupaj]+ [doslej skupno število odpravljenih napak vsake faze in skupaj iz zadnjega projektnega plana]
- [doslej % skupni odpravljenih napak vsake faze] = 100 * [doslej skupno število odpravljenih napak razvoja vsake faze] / [doslej skupno število odpravljenih napak skupaj]

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 1/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	29. 3. 2005
Program:	PROGRAM 8	**Zap.št:**	24
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**	**Skupaj doslej:**
Min/LOC:	6.65	6.64	6.64
LOC/Uro:	9.03	9.04	9.04
Št.napak/KLOC:		58.82	56.91

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	112	102	246
Največja velikost:	167		
Najmanjša velikost:	88		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	58	42	116	7.10
Načrtovanje	60	57	134	8.20
Kodiranje	431	440	994	60.83
Pregled kode	0	45	45	2.75
Prevajanje	45	10	68	4.16
Testiranje	97	48	172	10.53
Analiza	54	35	105	6.43
Skupaj:	745	677	1634	100.00
Največji čas:	1111			
Najmanjši čas:	585			

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 2/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	29 3. 2005
Program:	PROGRAM 8	**Zap.št:**	24
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

VNEŠENE NAPAKE:	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				0.00
Načrtovanje		1	3	21.43
Kodiranje		5	11	78.57
Pregled kode				0.00
Prevajanje				0.00
Testiranje				0.00
Analiza				0.00
Skupaj:		6	14	100.00

ODPRAVLJENE NAPAKE:	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje				0.00
Načrtovanje				0.00
Kodiranje				0.00
Pregled kode		3	3	21.43
Prevajanje		2	8	57.14
Testiranje		1	3	21.43
Analiza				0.00
Skupaj:		6	14	100.00

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 1/2

Izvajalec: ANDREJ NOVAK Datum: 11. 4. 2005
Program: PROGRAM 9 Zap.št: 29
Vodja: PRED. ROŽANC Pr.jezik: JAVA

SKUPAJ:	Plan:	Dej.vrednost:	Skupaj doslej:
Min/LOC:	6.64		
LOC/Uro:	9.04		
Št.napak/KLOC:	56.91		

VELIKOST (LOC):	Plan:	Dej. vrednost:	Skupaj doslej:
Skupaj (nove+spr.):	205		
Največja velikost:	294		
Najmanjša velikost:	149		

ČAS RAZVOJA (min):	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje	97			
Načrtovanje	112			
Kodiranje	828			
Pregled kode	37			
Prevajanje	57			
Testiranje	143			
Analiza	87			
Skupaj:	1361			
Največji čas:	1952			
Najmanjši čas:	989			

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 2/2

Izvajalec: ANDREJ NOVAK Datum: 29 3. 2005
Program: PROGRAM 8 Zap.št: 24
Vodja: PRED. ROŽANC Pr.jezik: JAVA

VNEŠENE NAPAKE:	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje				
Načrtovanje	3			
Kodiranje	9			
Pregled kode				
Prevajanje				
Testiranje				
Analiza				
Skupaj:	12			

ODPRAVLJENE NAPAKE:	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje				
Načrtovanje				
Kodiranje				
Pregled kode	3(2)			
Prevajanje	7			
Testiranje	2(3)			
Analiza				
Skupaj:	12			

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 1/2

Izvajalec: ANDREJ NOVAK — Datum: 11. 4. 2005
Program: PROGRAM 9 — Zap.št: 29
Vodja: PRED. ROŽANC — Pr.jezik: JAVA

SKUPAJ:	Plan:	Dej.vrednost:	Skupaj doslej:
Min/LOC:	6.64	5.65	6.16
LOC/Uro:	9.04	10.63	9.74
Št.napak/KLOC:	56.91	38.46	47.92

VELIKOST (LOC):	Plan:	Dej. vrednost:	Skupaj doslej:
Skupaj (nove+spr.):	205	234	480
Največja velikost:	294		
Najmanjša velikost:	149		

ČAS RAZVOJA (min):	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje	97	90	206	6.97
Načrtovanje	112	120	254	8.60
Kodiranje	828	810	1804	61.05
Pregled kode	37	65	110	3.72
Prevajanje	57	50	118	3.99
Testiranje	143	110	282	9.54
Analiza	87	76	181	6.13
Skupaj:	1361	1321	2955	100.00
Največji čas:	1952			
Najmanjši čas:	989			

PSP: Planiranje napak

PROJEKTNI PLAN (verzija 5) – 2/2

Izvajalec: ANDREJ NOVAK — Datum: 29 3. 2005
Program: PROGRAM 8 — Zap.št: 24
Vodja: PRED. ROŽANC — Pr.jezik: JAVA

VNEŠENE NAPAKE:	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje				0.00
Načrtovanje	3	4	7	30.43
Kodiranje	9	5	16	69.57
Pregled kode				0.00
Prevajanje				0.00
Testiranje				0.00
Analiza				0.00
Skupaj:	12	9	23	100.00

ODPRAVLJENE NAPAKE:	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje				0.00
Načrtovanje				0.00
Kodiranje				0.00
Pregled kode	3(2)	6	9	39.13
Prevajanje	7	2	10	43.48
Testiranje	2(3)	1	4	17.39
Analiza				0.00
Skupaj:	12	9	23	100.00

PSP: Odstranjevanje napak

Odstranjevanje napak je neposredno povezano s:

- stroški,
- časom razvoja in
- kakovostjo izdelka

Kakovostno delo

- navkljub razvoju tehnologije se narava dela razvijalcev PO v zadnjih 30 letih ni bistveno spremenila
- obstajajo boljši pristopi, vendar se v praksi ne uporabljajo pogosto

Problem odstranjevanja napak

- kompleksnost PO naraste za 10 krat vsakih 10 let
- temu primerno je zahtevnejše tudi obvladovanje napak
- potreba po spremljanju učinkovitosti odkrivanja in odpravljanja napak

PSP: Odstranjevanje napak

Dva možna pristopa za spremljanje učinkovitosti obvladovnja napak:

- število odpravljenih napak/ uro
- delež odpravljenih napak (yield)

Čas odpravljanja napak

- zahtevnejši izdelki - večji relativni delež vnešenih napak
- zahtevnejši izdelki - večji čas za odpravo ene napake
- praksa kaže, da se s tem tudi zmanjšuje delež odkritih (odpravljenih) napak

Primer: študenti pišejo manjše programe v C-ju

Vnesejo:

- 1 - 3 napak/uro med načrtovanjem
- 5 - 8 napak/uro med kodiranjem

Odpravijo:

- 2 - 4 napak/uro pri testiranju
- 6 - 12 napak/uro med pregledovanjem kode

PSP: Odstranjevanje napak

Tipične vrednosti števila odkritih napak:

- glede na uporabo PSP
- glede na velikost izdelka
- glede na faze procesa

NAPAKE:	500 brez PSP	500 z PSP	10K brez PSP	10K z PSP
Skupaj napak:	50	25	1000	500
Odkritih v Pregledu kode	0	15	0	300
Preostalih napak:	50	10	1000	200
Odkritih v Prevajanju	25	5	500	100
Odkritih v Testiranju	20	4	400	80
Preostalih napak:	5	1	100	20

PSP: Odstranjevanje napak

Tipične vrednosti časa odpravljanja napak:

- glede na uporabo PSP
- glede na velikost izdelka
- glede na faze procesa (potrebna testiranja)

ČAS ODPRAVE(ure)	500 brez PSP	500 z PSP	10K brez PSP	10K z PSP
Čas pregledovanja kode	0	2.5	0	50
Čas prevajanja	2	0.5	40	10
Čas testiranja modulov	10	2	200	40
Odstranjevanje napak	12	5	240	100
Čas testiranja združitve			500	100
Čas testiranja sistema			500	100
SKUPAJ	12	5	1240	300

PSP: Odstranjevanje napak

Določanje vrednosti Število napak/Uro:

- določamo samo globalne vrednosti - za vse projekte doslej
- določamo za vnešene in odpravljene napake
- določamo samo za določene faze procesa:
  - vnos napak: Načrtovanje in Kodiranje
  - odprava napak: Pregled kode, Prevajanje in Testiranje

[Doslej skupaj št.vnešenih/odpravljenih napak/Uro v izbrani fazi] =
60 *
[Doslej skupno število vnešenih/odpravljenih napak v izbrani fazi] /
[Doslej skupen čas v izbrani fazi]

PSP: Odstranjevanje napak

Določanje Deleža odpravljenih napak (pred prevajanjem) - yield:

- ključno vprašanje: kolikšen naj bo obseg testiranja?
- razmerje med napakami, ki jih odkrijemo pred prevajanjem (s pregledom kode) in vsemi vnešenimi napakami (odkritimi tudi pri prevajanju in testiranju)
- prve so veliko "cenejše" kot druge
- šteje število napak, ki so vnešene pred prevajanjem
- delež odpravljenih napak planiramo in merimo, določamo pa tudi globalno vrednost - za vse projekte doslej

[Delež odpravljenih napak pred prevajanjem] =
100 *
[skupno število odpravljenih napak pred prevajanjem] /
[skupno število vnešenih napak pred prevajanjem]

PSP: Odstranjevanje napak

Vse vrednosti obeh vrst podatkov vnesemo v Plan projekta:

- nova verzija: 6
- dopolnjeno Navodilo za izpolnjevanje plana projekta
- Pred izvedbo:
  - planiramo delež odpravljenih napak
- Po izvedbi:
  - vnesemo dejanski delež odpravljenih napak skupaj in skupaj doslej
  - vnesemo vsa števila napak/uro po ustreznih fazah

Primer: * Podatki niso čisto konsistentni; podatki o velikosti in času se nanašajo na več programov, napake pa smo spremljali le pri dveh!

Navodilo za izvajanje procesa se ne spremeni !

PSP: Odstranjevanje napak

Navodilo za izpolnjevanje projektnega plana - 6 (1/9)

Namen: Obrazec vsebuje planirane in dejansko izmerjene podatke o projektu v čitljivi in uporabni obliki

Glava: Vnesite:

- ime razvijalca in datum začetka
- naziv in zaporedno številko programa
- ime vodje in programski jezik, v katerem bo napisan program

Min/LOC: Pred razvojem:

- vnesite planirano vrednost [Min/LOC]. Uporabite doslej skupni Min/LOC iz zadnjega projektnega plana ali beležke opravil.

Po razvoju:

- dejanski [Min/LOC] = [izmerjen čas] / [izmerjena velikost]
- doslej skupni [Min/LOC] = [doslej skupni izmerjen čas] / [doslej skupna izmerjena velikost]

PSP: Odstranjevanje napak

Navodilo za izpolnjevanje projektnega plana – 6 (2/9)

LOC/Uro: Pred razvojem:

- planiran [LOC/Uro] = 60 / [Min/LOC]

Po razvoju:

- dejanski [LOC/Uro] = 60 / [Min/LOC]
- doslej skupni [LOC/Uro] = 60 / doslej skupni [Min/LOC]

Št.napak/KLOC: Pred razvojem:

- planirano [Št.napak/KLOC] = [Doslej skupaj št.napak/KLOC] iz zadnjega projektnega plana

Po razvoju:

- dejansko [Št.napak/KLOC] = 1000 * [dejansko število napak skupaj] / [dejanska velikost]
- [Doslej skupaj št.napak/KLOC] = 1000 * [doslej skupaj število napak] / [doslej skupna velikost]

PSP: Odstanjevanje napak

Navodilo za izpolnjevanje projektnega plana – 6 (3/9)

Delež odst.napak: Pred razvojem:

- planiran [Delež odpravljenih napak pred prevajanjem] = 100 * planirano [skupno število odpravljenih napak pred prevajanjem] / planirano [skupno število vnešenih napak pred prevajanjem]

Po razvoju:

- dejanski [Delež odpravljenih napak pred prevajanjem] = 100 * dejansko [skupno število odpravljenih napak pred prevajanjem] / dejansko [skupno število vnešenih napak pred prevajanjem]
- doslej skupen [Delež odpravljenih napak pred prevajanjem] = 100 * doslej skupaj [število odpravljenih napak pred prevajanjem] / doslej skupaj [skupno število vnešenih napak pred prevajanjem]

Navodilo za izpolnjevanje projektnega plana – 6 (4/9)

Velikost (LOC): Pred razvojem:

- vnesite planirane vrednosti za skupno, največjo in najmanjšo velikost nove in spremenjene kode

Po razvoju:

- seštejte dejanske vrstice nove in spremenjene kode
- [doslej skupna velikost] = [dejanska velikost]+[doslej skupna velikost zadnjega] projektnega plana

Čas razvoja (min): Pred razvojem:

- določite planirane vrednosti časa razvoja po naslednjih formulah:
  [planiran skupni čas] = [planirana skupna velikost] * [Min/LOC]
  [planiran največji čas] = [planirana največja velikost] * [Min/LOC]
  [planiran najmanjši čas] = [planirana najmanjša velikost] * [Min/LOC]

Navodilo za izpolnjevanje projektnega plana – 6 (5/9)

Čas razvoja (min): Pred razvojem (nadaljevanje):

- poiščite [doslej %] vrednosti za vsako fazo posebej iz zadnjega projektnega plana
- izračunajte planirane čase razvoja na naslednji način:
  [planiran čas faze] = [planiran skupni čas] * [doslej % faze] / 100

Po razvoju:

- izmerite dejanski čas razvoja vsake faze in skupaj
  (s pomočjo Beležke porabe časa)
- [doslej skupni čas razvoja vsake faze in skupaj] =
  [dejanski čas vsake faze in skupaj] +
  [doslej skupen čas razvoja vsake faze in skupaj iz zad.projektnega plana]
- [doslej % skupni čas razvoja vsake faze] =
  100 * [doslej skupen čas razvoja vsake faze] /
  [doslej skupni čas razvoja skupaj]

Navodilo za izpolnjevanje projektnega plana – 6 (6/9)

Vnešene napake: Pred razvojem:

- določite planirano število vnešenih napak (po fazah in skupaj)
- planirano [število vnešenih napak skupaj] = planirano [Št.napak/KLOC] * planirana [skupno število vrstic nove in spremenjene kode] / 1000
- poiščite [doslej %] števila vnešenih napak za vsako fazo posebej iz zadnjega projektnega plana
- planirano [število vnešenih napak faze] = planirano [skupno število vnešenih napak] * [doslej % vnešenih napak faze] / 100

Po razvoju:

- preštejte dejansko število vnešenih napak v vsaki fazi in skupaj (s pomočjo Beležke napak)
- [doslej skupno število vnešenih napak vsake faze in skupaj] = [dejansko število vnešenih napak vsake faze in skupaj]+ [doslej skupno število vnešenih napak vsake faze in skupaj iz zad.pr.plana]

Navodilo za izpolnjevanje projektnega plana – 6 (7/9)

Vnešene napake: Po razvoju (nadaljevanje):

- [doslej % skupni vnešenih napak vsake faze] = 100 * [doslej skupno število vnešenih napak razvoja vsake faze] / [doslej skupno število vnešenih napak skupaj]
- [doslej skupaj število vnešenih napak/Uro v Načrtovanju] = 60 * [doslej skupno število vnešenih napak v Načrtovanju] / [doslej skupen čas v fazi Načrtovanja]
- [doslej skupaj število vnešenih napak/Uro v Kodiranju] = 60 * [doslej skupno število vnešenih napak v Kodiranju] / [doslej skupen čas v fazi Kodiranja]

PSP: Odstranjevanje napak

Navodilo za izpolnjevanje projektnega plana – 6 (8/9)

Odpravljene napake: Pred razvojem:

- določite planirano število odpravljenih napak (po fazah in skupaj)
- planirano [število odpravljenih napak skupaj] =
  planirano [število vnešenih napak skupaj]
- poiščite [doslej %] števila odpravljenih napak za vsako fazo posebej
  iz zadnjega projektnega plana
- planirano [število odpravljenih napak faze] =
  planirano [skupno število odpravljenih napak] *
  [doslej % odpravljenih napak faze] / 100

PSP: Odstranjevanje napak

Navodilo za izpolnjevanje projektnega plana – 6 (9/9)

Odpravljene napake: Po razvoju:

- preštejte dejansko število odpravljenih napak v vsaki fazi in skupaj
  (s pomočjo Beležke napak)
- [doslej skupno število odpravljenih napak vsake faze in skupaj] =
  [dejansko število odpravljenih napak vsake faze in skupaj] +
  [doslej skupno število odpravljenih napak vsake faze in skupaj iz
  zadnjega projektnega plana]
- [doslej % skupni odpravljenih napak vsake faze] = 100 *
  [doslej skupno število odpravljenih napak razvoja vsake faze] /
  [doslej skupno število odpravljenih napak skupaj]
- [doslej skupaj število odpravljenih napak/Uro] v
  Pregledovanju kode / Prevajanju / Testiranju = 60 *
  [doslej skupno število odpravljenih napak] v
  Pregledovanju kode / Prevajanju / Testiranju /
  [doslej skupen čas] v Pregledovanju kode / Prevajanju / Testiranju

PSP: Odstranjevanje napak

PROJEKTNI PLAN (verzija 6) – 1/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	11. 4. 2005
Program:	PROGRAM 9	**Zap.št:**	29
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

SKUPAJ:	**Plan:**	**Dej.vrednost:**	**Skupaj doslej:**
Min/LOC:	6.64	5.65	6.16
LOC/Uro:	9.04	10.63	9.74
Št.napak/KLOC:	56.91	38.46	47.92
Delež odprav. napak:	25.00*	66.67*	39.13*

VELIKOST (LOC):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**
Skupaj (nove+spr.):	205	234	480
Največja velikost:	294		
Najmanjša velikost:	149		

ČAS RAZVOJA (min):	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**
Planiranje	97	90	206	6.97
Načrtovanje	112	120	254	8.60
Kodiranje	828	810	1804	61.05
Pregled kode	37	65	110	3.72
Prevajanje	57	50	118	3.99
Testiranje	143	110	282	9.54
Analiza	87	76	181	6.13
Skupaj:	1361	1321	2955	100.00
Največji čas:	1952			
Najmanjši čas:	989			

PSP: Odstranjevanje napak

PROJEKTNI PLAN (verzija 6) – 2/2

Izvajalec:	ANDREJ NOVAK	**Datum:**	29 3. 2005
Program:	PROGRAM 8	**Zap.št:**	24
Vodja:	PRED. ROŽANC	**Pr.jezik:**	JAVA

VNEŠENE NAPAKE:	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**	**Doslej Nap./Uro:**
Planiranje				0.00	
Načrtovanje	3	4	7	30.43	1.65*
Kodiranje	9	5	16	69.57	0.53*
Pregled kode				0.00	
Prevajanje				0.00	
Testiranje				0.00	
Analiza				0.00	
Skupaj:	12	9	23	100.00	

ODPRAVLJENE NAPAKE:	**Plan:**	**Dej. vrednost:**	**Skupaj doslej:**	**Doslej %:**	**Doslej Nap./Uro:**
Planiranje				0.00	
Načrtovanje				0.00	
Kodiranje				0.00	
Pregled kode	3(2)	6	9	39.13	4.91*
Prevajanje	7	2	10	43.48	5.08*
Testiranje	2(3)	1	4	17.39	0.85*
Analiza				0.00	
Skupaj:	12	9	23	100.00	

PSP: Odstranjevanje napak

Načini za izboljšanje učinkovitosti odprave napak:

- osredotočimo se na cilj, da (čimprej) odstranimo čimveč napak
- obvezno izvajamo preglede kode
- pri tem uporabljajmo opomnik, ki ga sproti prilagajamo ključnim vrstam odkritih napak
- smiselno presojajmo postopke: ni učinkovito veliko delati, če ni rezultatov

Načini za zmanjšanje vnašanja napak:

- beležimo vse odkrite napake
- poskušamo bolje načrtovati: preprečimo napake zaradi slabo definiranega načrta in olajšamo kodiranje
- uporabimo boljše metode in pristope
- uporabimo boljša orodja

PSP: Kakovost izdelka

Zagotavljanje kakovosti izdelka

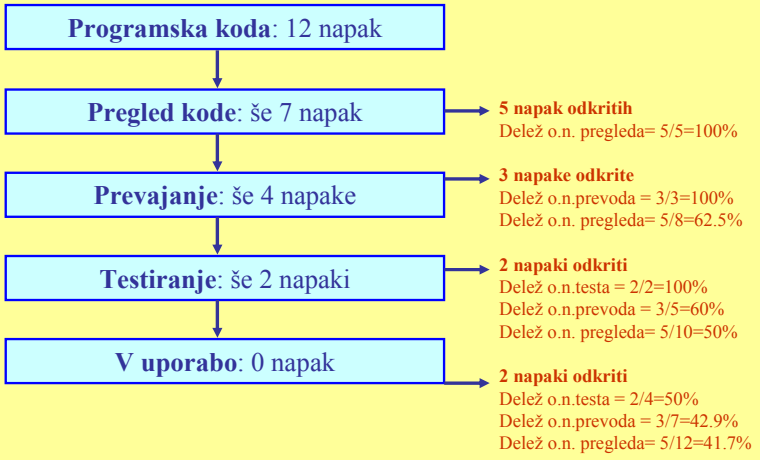
- zagotavljanje kakovosti je drago
- prednosti pregledovanja kode pred testiranjem: vgrajujemo kakovost
- kakovost vgrajujemo sproti, z testiranjem jo le preverjamo
- testiranje z zahtevnostjo postaja vse dražje in manj učinkovito
- na različne pristope za odpravljanje napak lahko gledamo kot na množico filtrov – vsak izloči del napak:
  - pregled kode: 70-80%
  - pregled sodelavca: 50-70%
  - prevajanje: 50%
  - testiranje modula: 40-50%
  - testiranje ob združevanju modulov: 45%
  - testiranje zahtev: 45%
  - Testiranje algoritmov: 8%

PSP: Kakovost izdelka

Določanje deležev odstranjenih napak

- bolj natančno – po vsaki fazi
- omogoča oceno, koliko napak bo ostalo v izdelku, ko bo ta končan
- cilj: želimo odkriti vse napake
- to lahko realno ugotovimo samo, če merimo med uporabo
- ocena na podlagi števila odkritih napak v fazah, ki odpravljajo napake
- **Pravilo: v izdelku ostane toliko napak, kot smo jih odkrili v zadnji fazi** (isto kot 50% delež odkritih napak v zadnji fazi, v drugih tipično še manj)
- zgodovinski podatki omogočajo, da ocenjujemo odstranjevanje napak in ukrepamo, da zagotovimo ustrezno raven z boljšimi pristopi
- 100% delež odkritih napak je zelo težko doseči, je pa mogoče s natančnim spremljanjem števila napak, prilagajanjem postopkov, izkušnjami, ...
- pomagamo si lahko s postopki, kot so prototipiranje ...

PSP: Kakovost izdelka



PSP: Kakovost procesa

Merjenje procesa

- ključna vloga procesa pri zagotavljanju kakovosti
- merjenje je nujno za ugotavljanje stanja procesa
- merimo:
  - velikost (izdelkov)
  - stroške (čas)
  - kakovost
    a) izdelkov: število napak, odstranjevanje napak, učinkovitost odpravljanja napak
    b) procesa: NAPAKE + **zahteve, načrt, performanse, vmesniki ...**

- kako ovrednotiti kakovost procesa ?

PSP: Kakovost procesa

Paradoks odstranjevanja napak

Učinkovitost odkrivanja napak se zmanjša, če se poveča kakovost izdelka

Logično v ekstremnih primerih:

- če ni napak, je učinkovitost odkrivanja napak nič
- če je napak zelo veliko, je tudi učinkovitost (kakršnegakoli) odkrivanja napak velika

Velja za preglede kode in testiranje, manj za prevajanje

Z naraščanjem kakovosti izdelkov:

- je v izdelkih vse manj napak
- je napake vse težje najti
- število napak v vsakem naslednjem koraku odkrivanja napak zelo pada

Posledica:

Čim več napak odkrijemo, tem bolj pomembno je odpraviti čisto vse napake.

PSP: Kakovost procesa

Veliki programi imajo dve vrsti napak:

- napake v posameznem modulu
- napake, ki nastanejo zaradi medsebojne povezanosti več modulov

Napake prve vrste:

- odpravimo z ustreznim pristopom (PSP)
- merimo z deležem odkritih napak

Napake druge vrste:

- so veliko zahtevnejše
- PSP – vpliva samo delno
- strategija odkrivanja napak

PSP: Kakovost procesa

Strategija odstranjevanja napak druge vrste:

a) posamezni moduli naj bodo kar najbolj kakovostni
b) izvesti je treba natančne preglede vmesnikov in komunikacije med moduli
c) preverimo, da so vse pomembne zahteve razumljene, načrtovane in izvedene
d) preverimo, da sistemski in programski načrt popolno vsebuje vse zahteve
e) izvedemo temeljito testiranje modulov po predhodnem pregledu kode
f) izvedemo temeljito združitveno testiranje
g) izvedemo temeljito sistemsko testiranje

Predpogoj: točka a)

PSP: Kakovost procesa

Cena kakovosti (cost of quality - COQ) :

- napake lahko odkrivamo zelo dolgo, a ne bomo odkrili vseh
- za inženirja je ključno, da zna uravnotežiti porabo časa in kakovost izdelka

Trije glavni elementi cene kakovosti:

a) cena odprave napak:
- cena odprave napak – popravljanja programa
- vključuje recimo tudi uporabo debuggerja, ponovno načrtovanje ipd.

b) cena iskanja napak:
- strošek odkrivanja napak (pregled kode, prevajanja, testiranja)
- brez odprave napak – kot bi preverjali program brez napak

c) cena preventivnih ukrepov
- recimo spremembe procesa z dodatno analizo, prototpiranje ipd.

PSP: Kakovost procesa

Izračun cene kakovosti v PSP-ju :

- cena iskanja napak: čas pregleda kode
- cena odprave napak: čas prevajanja in čas testiranja

Ceno kakovosti izražamo v odstotkih celotnega časa razvoja:

COQ iskanja napak = 100 *
čas pregleda kode /
celoten čas razvoja

COQ odprave napak = 100 *
(čas prevajanja + čas testiranja) /
celoten čas razvoja

PSP: Kakovost procesa

Primer:

Čas pregleda kode = 29 minut
Čas prevajanja = 5 minut
Čas testiranja = 10 minut
Skupen čas razvoja = 262 minut

COQ iskanja napak = 100 * 29 / 262 = 2900/ 262 = 11.07 %

COQ odprave napak = 100 * (5 + 10) / 262 = 1500/262 = 5.73 %

PSP: Kakovost procesa

Razmerje med COQ iskanja in odprave napak:

- meritev veliko pove o kakovosti procesa, če velja, da je delež odkritih napak primerno velik (nad 70%)
- PSP jo uporablja kot globalno meritev kakovosti procesa

Razmerje iskanje/odprava napak (Razmerje I/O napak) =
COQ iskanja napak / COQ odprave napak =
čas pregleda kode / (čas prevajanja + čas testirtanja)

Razmerje iskanja/odprave napak (Razmerje I/O napak) =
11.07% / 5.73% =
29 min / (5 min + 10 min) = 1.93

PSP: Kakovost procesa

Pomen razmerja iskanje/odprave napak:

1 ali manj:

- testiranje programa tipično najde veliko napak
- visoka gostota napak

2 ali več:

- testiranje odkrije malo (če sploh kaj) napak
- nizka gostota napak

CILJ - razmerje I/O napak naj bo 2 ali več:

- na podlagi zgodovinskih podatkov ugotovimo čas prevajanja in testiranja
- predvidimo dvakrat daljši čas za pregled kode
- pregled kode mora biti učinkovit (prilagajanje opomnika)
- povečujemo čas za pregled kode dokler stalno ne dosegamo deleža o.n. > 80

PSP: Kakovost procesa

Razmerje I/O napak vnesemo tudi v Plan projekta:

- verzija 7 (končna)
- tri vrednosti: planirana, trenutna in doslej skupna vrednost

Spremeni se tudi Navodilo za izpolnjevanje Plana projekta:

- izračun razmerja med iskanjem in odpravo napak
- pred in po razvoju

Navodilo za izvajanje procesa razvoja se ne spremeni

Primer: graf razmerja med gostoto napak in razmerjem I/O napak

PSP: Kakovost procesa

PROJEKTNI PLAN (verzija 7) – 1/2

Izvajalec: ANDREJ NOVAK — Datum: 11. 4. 2005
Program: PROGRAM 9 — Zap.št: 29
Vodja: PRED. ROŽANC — Pr.jezik: JAVA

SKUPAJ:	Plan:	Dej.vrednost:	Skupaj doslej:
Min/LOC:	6.64	5.65	6.16
LOC/Uro:	9.04	10.63	9.74
Št.napak/KLOC:	56.91	38.46	47.92
Delež odprav. napak:	25.00*	66.67*	39.13*
Razmerje I/O napak:	0.185 (0.37, 2)	0.406	0.275

VELIKOST (LOC):	Plan:	Dej. vrednost:	Skupaj doslej:
Skupaj (nove+spr.):	205	234	480
Največja velikost:	294		
Najmanjša velikost:	149		

ČAS RAZVOJA (min):	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:
Planiranje	97	90	206	6.97
Načrtovanje	112	120	254	8.60
Kodiranje	828	810	1804	61.05
Pregled kode	37 (74, 400)	65	110	3.72
Prevajanje	57	50	118	3.99
Testiranje	143	110	282	9.54
Analiza	87	76	181	6.13
Skupaj:	1361	1321	2955	100.00
Največji čas:	1952			
Najmanjši čas:	989			

PSP: Kakovost procesa

PROJEKTNI PLAN (verzija 7) – 2/2

Izvajalec: ANDREJ NOVAK — Datum: 29 3. 2005
Program: PROGRAM 8 — Zap.št: 24
Vodja: PRED. ROŽANC — Pr.jezik: JAVA

VNEŠENE NAPAKE:	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:	Doslej Nap./Uro:
Planiranje				0.00	
Načrtovanje	3	4	7	30.43	1.65*
Kodiranje	9	5	16	69.57	0.53*
Pregled kode				0.00	
Prevajanje				0.00	
Testiranje				0.00	
Analiza				0.00	
Skupaj:	12	9	23	100.00	

ODPRAVLJENE NAPAKE:	Plan:	Dej. vrednost:	Skupaj doslej:	Doslej %:	Doslej Nap./Uro:
Planiranje				0.00	
Načrtovanje				0.00	
Kodiranje				0.00	
Pregled kode	3(2)	6	9	39.13	4.91*
Prevajanje	7	2	10	43.48	5.08*
Testiranje	2(3)	1	4	17.39	0.85*
Analiza				0.00	
Skupaj:	12	9	23	100.00	

Navodilo za izpolnjevanje projektnega plana - 7 (1/10)

Namen: Obrazec vsebuje planirane in dejansko izmerjene podatke o projektu v čitljivi in uporabni obliki

Glava: Vnesite:

- ime razvijalca in datum začetka
- naziv in zaporedno številko programa
- ime vodje in programski jezik, v katerem bo napisan program

Min/LOC: Pred razvojem:

- vnesite planirano vrednost [Min/LOC]. Uporabite doslej skupni Min/LOC iz zadnjega projektnega plana ali beležke opravil.

Po razvoju:

- dejanski [Min/LOC] = [izmerjen čas] / [izmerjena velikost]
- doslej skupni [Min/LOC] = [doslej skupni izmerjen čas] / [doslej skupna izmerjena velikost]

Navodilo za izpolnjevanje projektnega plana – 7 (2/10)

LOC/Uro: Pred razvojem:

- planiran [LOC/Uro] = 60 / [Min/LOC]

Po razvoju:

- dejanski [LOC/Uro] = 60 / [Min/LOC]
- doslej skupni [LOC/Uro] = 60 / doslej skupni [Min/LOC]

Št.napak/KLOC: Pred razvojem:

- planirano [Št.napak/KLOC] = [Doslej skupaj št.napak/KLOC] iz zadnjega projektnega plana

Po razvoju:

- dejansko [Št.napak/KLOC] = 1000 * [dejansko število napak skupaj] / [dejanska velikost]
- [Doslej skupaj št.napak/KLOC] = 1000 * [doslej skupaj število napak] / [doslej skupna velikost]

Navodilo za izpolnjevanje projektnega plana – 7 (3/10)

Delež odst.napak: Pred razvojem:

- planiran [Delež odpravljenih napak pred prevajanjem] = 100 * planirano [skupno število odpravljenih napak pred prevajanjem] / planirano [skupno število vnešenih napak pred prevajanjem]

Po razvoju:

- dejanski [Delež odpravljenih napak pred prevajanjem] = 100 * dejansko [skupno število odpravljenih napak pred prevajanjem] / dejansko [skupno število vnešenih napak pred prevajanjem]
- doslej skupen [Delež odpravljenih napak pred prevajanjem] = 100 * doslej skupaj [število odpravljenih napak pred prevajanjem] / doslej skupaj [skupno število vnešenih napak pred prevajanjem]

Navodilo za izpolnjevanje projektnega plana – 7 (4/10)

Razmerje I/O napak: Pred razvojem:

- planirano [Razmerje I/O napak] = planiran [čas za preged kode] / (planiran [čas za prevajanje] + planiran [čas za testiranje])

Po razvoju:

- dejansko [Razmerje I/O napak] = dejanski [čas za pregled kode] / (dejanski [čas za prevajanje]+dejanski [čas za testiranje])
- doslej skupeno [Razmerje I/O napak] = doslej skupaj [čas za pregled kode] / (doslej skupaj [čas za prevajanje] + doslej skupaj [čas za testiranje])

Navodilo za izpolnjevanje projektnega plana – 7 (5/10)

Velikost (LOC): Pred razvojem:

- vnesite planirane vrednosti za skupno, največjo in najmanjšo velikost nove in spremenjene kode

Po razvoju:

- seštejte dejanske vrstice nove in spremenjene kode
- [doslej skupna velikost] = [dejanska velikost]+[doslej skupna velikost zadnjega] projektnega plana

Čas razvoja (min): Pred razvojem:

- določite planirane vrednosti časa razvoja po naslednjih formulah:
  [planiran skupni čas] = [planirana skupna velikost] * [Min/LOC]
  [planiran največji čas] = [planirana največja velikost] * [Min/LOC]
  [planiran najmanjši čas] = [planirana najmanjša velikost] * [Min/LOC]

Navodilo za izpolnjevanje projektnega plana – 7 (6/10)

Čas razvoja (min): Pred razvojem (nadaljevanje):

- poiščite [doslej %] vrednosti za vsako fazo posebej iz zadnjega projektnega plana
- izračunajte planirane čase razvoja na naslednji način:
  [planiran čas faze] = [planiran skupni čas] * [doslej % faze] / 100

Po razvoju:

- izmerite dejanski čas razvoja vsake faze in skupaj
  (s pomočjo Beležke porabe časa)
- [doslej skupni čas razvoja vsake faze in skupaj] =
  [dejanski čas vsake faze in skupaj] +
  [doslej skupen čas razvoja vsake faze in skupaj iz zad.projektnega plana]
- [doslej % skupni čas razvoja vsake faze] =
  100 * [doslej skupen čas razvoja vsake faze] /
  [doslej skupni čas razvoja skupaj]

Navodilo za izpolnjevanje projektnega plana – 7 (7/10)

Vnešene napake: Pred razvojem:

- določite planirano število vnešenih napak (po fazah in skupaj)
- planirano [število vnešenih napak skupaj] = planirano [Št.napak/KLOC] * planirana [skupno število vrstic nove in spremenjene kode] / 1000
- poiščite [doslej %] števila vnešenih napak za vsako fazo posebej iz zadnjega projektnega plana
- planirano [število vnešenih napak faze] = planirano [skupno število vnešenih napak] * [doslej % vnešenih napak faze] / 100

Po razvoju:

- preštejte dejansko število vnešenih napak v vsaki fazi in skupaj (s pomočjo Beležke napak)
- [doslej skupno število vnešenih napak vsake faze in skupaj] = [dejansko število vnešenih napak vsake faze in skupaj] + [doslej skupno število vnešenih napak vsake faze in skupaj iz zad.pr.plana]

Navodilo za izpolnjevanje projektnega plana – 7 (8/10)

Vnešene napake: Po razvoju (nadaljevanje):

- [doslej % skupni vnešenih napak vsake faze] = 100 * [doslej skupno število vnešenih napak razvoja vsake faze] / [doslej skupno število vnešenih napak skupaj]
- [doslej skupaj število vnešenih napak/Uro v Načrtovanju] = 60 * [doslej skupno število vnešenih napak v Načrtovanju] / [doslej skupen čas v fazi Načrtovanja]
- [doslej skupaj število vnešenih napak/Uro v Kodiranju] = 60 * [doslej skupno število vnešenih napak v Kodiranju] / [doslej skupen čas v fazi Kodiranja]

Navodilo za izpolnjevanje projektnega plana – 7 (9/10)

Odpravljene napake: Pred razvojem:

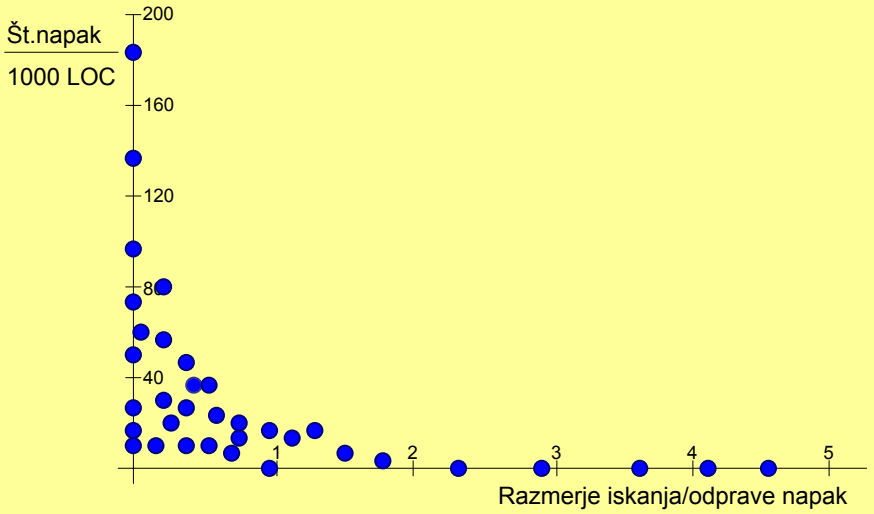
- določite planirano število odpravljenih napak (po fazah in skupaj)
- planirano [število odpravljenih napak skupaj] =
  planirano [število vnešenih napak skupaj]
- poiščite [doslej %] števila odpravljenih napak za vsako fazo posebej
  iz zadnjega projektnega plana
- planirano [število odpravljenih napak faze] =
  planirano [skupno število odpravljenih napak] *
  [doslej % odpravljenih napak faze] / 100

Navodilo za izpolnjevanje projektnega plana – 7 (10/10)

Odpravljene napake: Po razvoju:

- preštejte dejansko število odpravljenih napak v vsaki fazi in skupaj
  (s pomočjo Beležke napak)
- [doslej skupno število odpravljenih napak vsake faze in skupaj] =
  [dejansko število odpravljenih napak vsake faze in skupaj] +
  [doslej skupno število odpravljenih napak vsake faze in skupaj iz
  zadnjega projektnega plana]
- [doslej % skupni odpravljenih napak vsake faze] = 100 *
  [doslej skupno število odpravljenih napak razvoja vsake faze] /
  [doslej skupno število odpravljenih napak skupaj]
- [doslej skupaj število odpravljenih napak/Uro] v
  Pregledovanju kode / Prevajanju / Testiranju = 60 *
  [doslej skupno število odpravljenih napak] v
  Pregledovanju kode / Prevajanju / Testiranju /
  [doslej skupen čas] v Pregledovanju kode / Prevajanju / Testiranju

PSP: Kakovost procesa



PSP: Kakovost procesa

Opisan primer primeren za PSP, večji projekti zahtevajo natančnejše merjenje:

- vsaka od treh faz razvoja ima čas iskanja in čas odprave napak
- vse lahko razberemo iz beležke napak

Recimo:

Čas pregleda kode = 19 minut
Čas prevajanja = 4 minut
Čas testiranja = 10 minut
Skupen čas razvoja = 262 minut

Število napak: 2 Čas odprave:10 minut
Število napak: 1 Čas odprave:1 minut
Število napak: 0 Čas odprave:0 minut

COQ iskanja napak = 100 *(19+4+10) / 262 = 3300/262 = 12.595%
COQ odprave napak = 100 *(10+1+0) / 262 = 1100/262 = 4.198%

Razmerje I/O napak= 12.595% / 4.198% = (19+4+10) / (10+1+0) = 3.00

PSP: Kakovost procesa

Ime: Andrej Novak
Vodja: Pred. Rožanc

Datum: 11.04.2005
Št.prog: 9

Datum	Št.nap.	Kategorija	Faza vnosa	Faza odprave	Čas odprave	Št.pop.nap.
18.4.	1	20	kodiranje	pregled kode	2	
Opis: manjkajoče ; # 21						
18.4.	2	40	načrtovanje	pregled kode	8	
Opis: napačen tip (int namesto float) – spremenljivka a # 5						
18.4.	3	20	kodiranje	prevajanje	1	
Opis: napačno ime spremenljivke (velike – male črke) – spremenljivka stLeta # 9						
Opis:						

PSP: Kakovost procesa

Osebna zaveza za kakovosten razvoj programske opreme:

- pomen kakovosti
- tveganja slabe kakovosti
- zaveza za kakovost
- osebni cilji in kakovost
- nagrada za dosežek

Literatura

1. Watts S. Humphrey: Introduction to the Personal Software Process, Carnegie Mellon University, Addison-Wesley, 1997.

2. Watts S. Humphrey: A Discipline for Software Process, Carnegie Mellon University, Addison-Wesley, 1995.

3. Domača stran PSP: www.sei.cmu.edu/psp