Poučevanje kakovosti programske opreme s poudarkom na modelu PSP

Igor Rožanc, Viljan Mahnič

Fakulteta za računalništvo in informatiko, Tržaška 25, 1000 Ljubljana
igor.rozanc@fri.uni-lj.si, viljan.mahnic@fri.uni-lj.si

Kakovost programske opreme (PO) je pomembno področje, ki ga morajo na ustrezen način spoznati vsi študentje računalništva. Na naši fakulteti ga na visokem strokovnem študiju Računalništva in informatike poučujemo v okviru predmeta Razvoj programskih sistemov II (RPS II), ki se izvaja v 3. letniku. V prispevku najprej utemeljimo izbiro snovi za ustrezno predstavitev področja kakovosti PO. Primeren način za celovito predstavitev je obravnava uveljavljenega modela za zagotavljanje kakovosti. Model zrelostnih stopenj (CMM) je tak primer, vendar je primeren predvsem za velike organizacije. Za slovenske razmere je primernejši soroden model osebnega procesa razvoja (PSP), ki je namenjen inženirju-razvijalcu PO v organizaciji za razvoj PO. V nadaljevanju članka prikažemo glavne značilnosti PSP-ja, nakar predstavimo izvajanje predavanj in vaj pri predmetu RPS II. Čeprav predmet pokriva tudi druge tematike iz področja tehnologije PO, je osrednja vloga na predavanjih namenjena PSP-ju, na vajah pa študenti praktično uporabljajo PSP pri razvoju manjših spletnih aplikacij. Pri tem izdelajo več PSP dokumentov, na podlagi katerih smo izvedli analizo uspešnosti učenja PSP-ja. Pokazalo se je, da so študentje uspešno spoznali PSP, vendar učinkovita uporaba zahteva disciplino in postopno uvajanje. Na koncu predstavimo še rezultate anonimne mnenjske ankete med študenti, ki pokaže, da je po mnenju večine PSP priljubljen in zelo uporaben model.

Ključne besede: *tehnologija PO, kakovost PO, modeli za zagotavljanje kakovosti, model stopenj zrelosti (CMM), osebni proces razvoja PO (PSP), anketa*

Teaching Software Quality With Emphasis on PSP

Learning about software quality is a must for any computer science student. At Faculty of Computer and Information Science it is taught at course named Software Development II, which undergraduate students take in their third year. In article we present a discussion on selection of an appropriate quality model to cover software quality area first. Capability Maturity Model (CMM) is a well known and complete model, but it is useful for bigger computer companies mostly. In Slovene case it is better to choose Personal Software Process (PSP) model, which defines process for engineer in software organization. After that the basic characteristics of PSP are presented, and teaching process of Software Development II course explained. Course covers several other Software Engineering themes, but we dedicate our main attention to PSP. A practical exercise is part of course as well, and there students develop a small web application by the PSP principles. Some PSP documents are produced in this way, and those we analyzed at the end of course. We discovered PSP learning was successful, but efficient use demands additional gradual implementation of PSP principles. In last part we present the analysis of anonymous student questionnaire. Results proved that students mark PSP as a useful and practical model.

Key words: *Software Engineering, Software Quality, Quality Models, Capability Maturity Model (CMM), Personal Software Process (PSP), questionnaire*

1. Uvod

Učitelji naše fakultete se trudimo študente dobro oboroţiti z znanji, ki so specifična za naše področje. Med ta sodijo tudi znanja s področja **kakovosti PO**, saj kakovost po mnenju mnogih predstavlja ključni dejavnik uspeha pri razvoju PO (Paulk et al, 1993).

Poučevanje kakovosti PO je smiselno predvsem v višjih letnikih, ko imajo študenti že nekaj programerskih izkušenj. Na visokošolskem študiju Računalništvo in informatika smo to snov dodali k vsebini predmeta **RPS II**, ki se predava v (zadnjem) 3. letniku smeri Programska oprema. Taka odločitev je ustrezna, saj ta predmet pokriva področje **tehnologije PO** (ang. Software Engineering) in kot tak obravnava pristope, tehnike in orodja za *obdelavo zahtev (ang. requirements engineering)*, *načrtovanje PO (ang. design), izdelavo PO (ang. software development), preverjanje ustreznosti in pravilnosti PO (ang. validation and verification) ter vodenje razvoja PO (ang. management)* (Sommerville, 2004).

Namen prispevka je predstaviti naš pristop in izkušnje s poučevanjem kakovosti PO v okviru predmeta RPS II. V nadaljevanju bomo tako najprej prestavili izbiro tematike s področja kakovosti PO, ki bo študentom dovolj celovito in zanimivo predstavila področje. Sledila bo kratka predstavitev vsebine, načina poučevanja ter slednjič še rezultatov, ki smo jih dosegli pri tem. Slednje bomo ovrednotili tako z vidika učinka učenja (ocena izdelkov študentov) kot mnenj študentov, ki so jih izrazili v anketi.

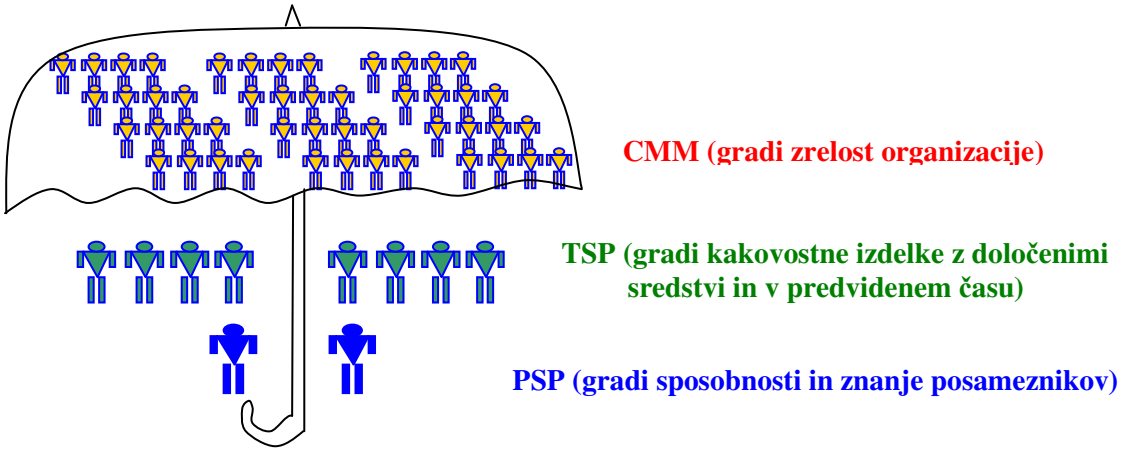
2. Izbira tematike s področja kakovosti PO

Za kakovosten programski izdelek ni dovolj, da pred zaključkom njegovega razvoja izločimo odkrite napake, temveč moramo med celotnim razvojem skrbeti za to, da bo napak čim manj. Za to potrebujemo učinkovito vgraditev aktivnosti za zagotavljanje kakovosti v celoten proces razvoja PO (Sommerville, 2004; Paulk et al, 1993). Z drugimi besedami to pomeni, da je kakovost PO neposredna posledica kakovostnega oziroma dobro **definiranega procesa razvoja PO**. Na tem izhodišču je bilo razvitih več standardov - recimo druţina standardov ISO 9000 (ISO, 2004a; ISO, 2004b)- in modelov za zagotavljanje kakovosti - recimo model BOOTSTRAP (Kuvaja et al,1994; Haase et al, 1994), model ISO/SPICE (Amam et al, 1998) ali model stopenj zrelosti - CMM (Paulk et al, 1993; SEI, 2002).

Za predstavitev študentom so zanimivi predvsem celoviti modeli, ki pokrivajo različne vidike zagotavljanja kakovosti. Eden primernih je recimo zelo uveljavljeni **model stopenj zrelosti** (ang. Capability Maturity Model ali CMM) (Paulk et al, 1993; SEI, 2002), ki omogoča tako ovrednotenje procesa razvoja PO kot izboljševanje kakovosti le-tega. Odlikuje se po natančni definiciji koncepta *petih stopenj zrelosti* (angl. maturity levels) razvojnega procesa. Stopnja zrelosti predstavlja nivo sposobnosti organizacije za razvoj PO. V praksi to pomeni, da ima le-ta tako dorečen proces razvoja PO, da dosega vse cilje ključnih področij, ki jih model pripisuje tej in nižjim stopnjam. Cilj organizacije je postopoma »zoreti«, kar pomeni ustrezno nadgrajevati svoj proces razvoja PO in vanj vgrajevati kakovost.

Vendar je model CMM (podobno kot drugi našteti modeli) namenjen predvsem velikim organizacijam za razvoj PO in je kot tak preobsežen in prezahteven za uporabo v manjših organizacijah, kakršnih je v Sloveniji velika večina. Kljub celovitosti in zanimivim konceptom tako podrobno spoznavanje CMM-ja (ali podobnega modela) ni najboljša rešitev. Bolje je CMM in ostale modele študentom predstaviti pregledno (brez podrobnosti), večino časa pa posvetiti predstavitvi problematike na bolj neposreden način, ki ga lahko študentje tudi sami preizkusijo pri praktičnih vajah. Posebej primeren se nam zdi model procesa, ki ni namenjen organizaciji za razvoj PO, temveč posamezniku, ki v takšni organizaciji dela.

Tak model so razvili avtorji CMM-ja, ko so se zavedli težav z uporabo le-tega v majhnih organizacijah. V bistvu so razvili dva modela: najprej **osebni proces razvoja PO** (ang. Personal Software Process ali PSP) (Humphrey, 1995; Humphrey, 2000b), pozneje pa še **skupinski proces razvoja PO** (Team Software Process ali TSP) (Humphrey, 2000a; Humphrey, 2000c). PSP je model procesa za posamezne inženirje, TSP pa njegova nadgradnja, ki določa proces za učinkovito delovanje razvojne skupine. Kot kaže slika 1 sta oba modela dopolnitev CMM-ja, ki pripomore k učinkovitejšemu delu razvijalcev in skupin na ravni organizacije (Humphrey, 1999), vendar se lahko uporabljata tudi samostojno.



Slika 1: Razmerje med CMM, TSP in PSP

V okviru predmeta RPS II smo se tako odločili, da študentom najprej pregledno predstavimo področje zagotavljanja kakovosti procesa razvoja PO na ravni organizacije (modele CMM, ISO 9000, Bootstrap, ISO/SPICE) in razvojne skupine (model TSP). Nato večji del časa posvetimo modelu PSP, ki ga temeljito in na ustrezen način obravnavamo tako na predavanjih kot vajah.

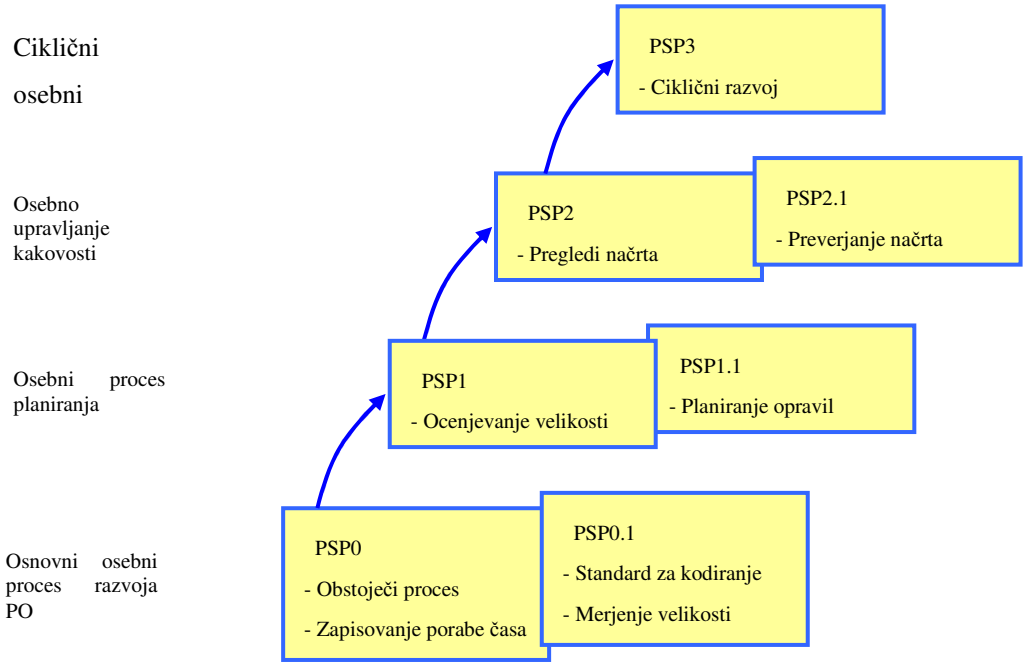
3. Osebni proces razvoja PO (PSP)

Osebni proces razvoja PO je bil razvit po letu 1995 na Software Engineering Institute, ki deluje v okviru ameriške univerze Carnegie Mellon (Humphrey, 1995; Humphrey, 2000b). Sestavlja ga množica metod, navodil in dokumentov o tem, kako učinkovito planirati, načrtovati, meriti in voditi potek dela posameznega inženirja, ki sodeluje pri razvoju PO. PSP je zastavljen splošno, saj pokriva opravila vseh faz razvoja PO in ni omejen z uporabo programskega jezika ali načina dela. Poleg tipičnih opravil v okviru razvoja PO ga lahko učinkovito uporabimo tudi pri drugih opravilih inženirja. V vseh primerih so izkušnje z uporabo tega pristopa zelo ugodne.

Namen uporabe PSP-ja je pravočasen in poceni razvoj kakovostne PO. Pri tem izhajamo iz dejstva, da je vsak inženir različen in je tako pri svojem delu lahko učinkovit le, če skrbno načrtuje svoje delo s pomočjo svojih preteklih izkušenj. Izboljšanje učinkovitosti dela zahteva discipliniran proces razvoja PO s sprotnim merskim spremljanjem tako rezultatov dela kot vloženega napora. Inženirji izdelajo kakovostne končne izdelke, če se čutijo osebno zavezane in odgovorne za najprimernejši način dela, ki do tega vodi. Ker pa je najboljši način dela tisti, ki najhitreje in najceneje vodi do rezultata, zaseda obvladovanje napak pomembno mesto v PSP-

ju; zgodnje odkrivanje in odpravljanje napak je cenejše od kasnejšega, še ceneje je sploh preprečiti nastanek napak.

Osebni proces razvoja PO je nastal na temelju tistih obstoječih znanj in izkušenj skupin in organizacij za razvoj PO, ki jih je mogoče prilagoditi za osebno uporabo inženirjev. Modeli (recimo CMM, ki je izhodišče PSP-ja) celovito in z različnih vidikov pokrivajo razvoj PO na ravni organizacije, kar je na ravni posameznika tudi naloga PSP-ja. S tega stališča bi PSP lahko opredelili tudi kot podmožico primernih metod in praks CMM-ja, ki so jih prilagodili ter sestavili v model. V skladu s principi CMM-ja inženir svoj delovni proces postopoma nadgrajuje, pri čemer začne z osnovnimi postopki in konča s tistimi, ki mu omogočajo popolno obvladovanje vseh vidikov svojega dela.



Slika 2: Sedem verzij modela PSP

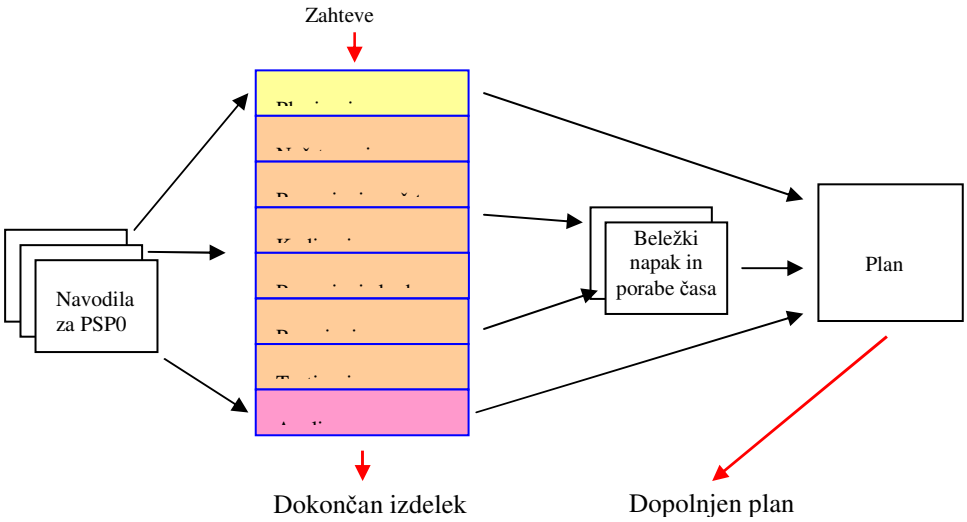
Slika 2 prikazuje sedem verzij modela PSP, ki predstavljajo osebne procese od najbolj enostavnega (PSP0) do najbolj popolnega (PSP3). Vsaka verzija obsega več področij, pri čemer velja, da vsaka višja verzija vsebuje tudi vsa področja nižje verzije. Inženir si bo na začetku za cilj izbral proces PSP0. Ko bo operativno obvladal vsa področja te verzije, bo njegov cilj PSP0.1, ki ga bo dosegel z izpolnitvijo dodatnih zahtev. V smislu CMM-ja bi lahko verzije razvrstili na 4 (zrelostne) stopnje:

1. **stopnja** obsega verziji PSP0 in PSP0.1. PSP0 vzpostavlja nadziran osebni proces inženirja z osnovnimi merjenji in beleženjem rezultatov, ki je nujen temelj za kakršenkoli napredek. PSP0.1 nadgrajuje PSP0 z uvajanjem standarda za kodiranje, merjenjem velikosti izdelkov in beleženjem možnosti za izboljšavo.
2. **stopnja** govori o planiranju. PSP1 tako uvaja poročanje o testiranju ter planiranje velikosti in porabe virov, PSP1.1 pa planiranje opravil in porabe časa. Veliko je govora o metodah za ocenjevanje velikosti (recimo PROBE (Humphrey, 1995)).

3. **stopnja** uvaja upravljanje s kakovostjo. PSP2 uvaja tehnike za pregledovanje načrtov in kode, ki omogočajo učinkovito odkrivanje napak in ocenjevanje kakovosti. PSP2.1 dodaja tehnike za preverjanje načrtov.
4. **stopnja** obsega eno verzijo (PSP3), ki dodaja tehnike za obvladovanje razvoja večjih programov s cikličnim ponavljanjem PSP2 na delih programov.

Opreratívno je vsaka verzija PSP-ja definirana z množico dokumentov štirih vrst:

- **Navodilo** (ang. script) zajema opis zaporedja vseh korakov, ki predstavljajo proces (recimo PSP0) ali del procesa (recimo planiranje v PSP0). Koraki so v enem stavku opisane aktivnosti, njihovo zaporedje pa služi določenemu namenu, kar je iz navodila tudi razvidno. Vsako navodilo ima definirane tudi vhodne in izhodne kriterije.
- **Obrazci** (ang. forms) pomagajo pri zbiranju in zapisovanju rezultatov v natanko določeni obliki. Ker natanko določajo število in vrsto podatkov, služijo kot predloga in opomnik, kaj je treba opraviti.
- **Beležke** (ang. log) so nekoliko podobne obrazcem, a vsebujejo manj podatkov in običajno služijo za zapis izmerjenih podatkov, ki se lahko pojavijo velikokrat.
- **Standardi** (ang. standards) sistematično opisujejo pravila z jasnim opisom, tipičnimi vrednostmi, primeri, izjemami in podobnim.



Slika 3: Primer uporabe osebnega procesa PSP0

Primer uporabe osebnega procesa razvoja PO (verzija PSP0) je prikazan na sliki 3, na kateri lahko zasledimo tri vrste opisanih dokumentov. V skladu s strukturo PSP0 je inženirjev prvi korak pri razvoju PO (recimo izdelavi programa v javi) **planiranje** (ang. planning) na podlagi definiranih **zahtev** (ang. requirements). To izvede v skladu z družino **navodil za izvajanje PSP0.** Rezultate planiranja (predvideno porabo časa in velikost izdelka) shrani v obrazec **plan** (ang. plan summary), nato pa razvija izdelek v skladu z izbranim načinom dela, ki ga tudi opisujejo navodila. V našem primeru ta vsebuje **načrtovanje** (ang. design), **kodiranje** (ang. coding), **pregled kode** (ang. coding review), **prevajanje** (ang. compile) in **testiranje** (ang. test). Če ta razvoj predstavimo z izdelki, inženir najprej izdela načrt rešitve (recimo določi strukturo programa, razrede, metode in atribute), v skladu z njim napiše programsko kodo (javanske razrede in metode), to še enkrat pregleda, prevede in na koncu še ustrezno testira. Vse

skupaj lahko traja uro ali več dni, vendar v vsakem primeru inženir sproti beleži podatke o porabljenem času in odkritih napakah v **beležki napak in porabe časa**. Na koncu inženir izvede še **analizo** (ang. postmortem), v kateri izmeri velikost izdelka. Velikost skupaj z izvlečkom podatkov iz **beležke napak in porabe časa** zapiše v **plan**, ki skupaj z **dokončanim izdelkom** (programom v javi) predstavlja končni rezultat dela.

Proces v kasnejših fazah bi lahko predstavili z dopolnitvijo slike 3 z novimi elementi. Za primerjavo zahtevnosti posameznih verzij (Humphrey, 1995) navaja, da je PSP0 definiran z 11 dokumenti (na sliki so vidni le glavni), pri PSP1 število doseže 19 dokumentov, pri PSP2 25 dokumentov in v primeru najbolj izdelanega PSP3 kar 39 elementov. Podrobnejši opis PSP-ja presega namen našega prispevka in si ga bralec lahko ogleda v (Humphrey, 1995 in 2000b).

4. Poučevanje PSP-ja v okviru predmeta RPS II

Za predmet RPS II so tedensko predvidene tri ure predavanj, ena ura avditornih in dve laboratorijskih vaj, ki se izvajajo v več ciklih. V tem času skušamo študentom vsebinsko predstaviti izbor tematik z različnih področij tehnologije PO (Sommerville, 2004). Pristope in tehnike za obdelavo zahtev, načrtovanje in izdelavo PO študentje spoznavajo na predavanjih s pregledom ene od tradicionalnih strukturnih metodologij SSADM (Goodland et al, 1995), sodobnejši pristopi pa so prikazani z agilnimi metodologijami Scrum (Schwaber, 2001) in ekstremno programiranje - XP (Beck, 2000). Kot rečeno je področje kakovosti PO na predavanjih pokrito s pregledom modelov in standardov za zagotavljanje kakovosti PO (predvsem CMM in TSP) ter podrobno predstavitvijo PSP-ja. Programske tehnike in orodja obravnavajo študentje v okviru vaj, ko spoznajo PL/SQL (Feuerstein, 2002), Oracle Developer (Koletzke et al, 1999) in Oracle Portal (Greenwald et al, 2001) ter pregledno še nekatere druge tehnologije za razvoj spletnih aplikacij.

V okviru prispevka nas še zlasti zanima poučevanje PSP-ja, ki mu na predavanjih namenjamo največ poudarka. Metodološko gledano smo v skladu s (Humphrey, 1997) snov nekoliko poenostavili in prilagodili uporabi PSP-ja pri razvoju manjših programov. PSP tako ni predstavljen po verzijah, temveč po vsebinskih sklopih:

- najprej obdelamo **obvladovanje časa** (spremljanje in napovedovanje porabe časa, določanje in prilagajanje urnikov),
- sledi **velikost izdelkov** (spremljanje in napovedovanje obsega izdelkov),
- **proces razvoja PO** (osnovni in nadgradnja),
- **kakovost izdelkov** (spremljanje in napovedovanje števila vnešenih in odpravljenih napak)
- in končno **kakovost procesa** (spremljanje in napovedovanje ustreznih pokazateljev kakovosti procesa).

Takšno zaporedje prikaza snovi je za študente bolj razumljivo, predvsem pa so zahtevnejši deli metode (recimo ocenjevanje velikosti programa po metodi PROBE (Humphrey, 1995)) samo nakazani, da študentje začutijo obseg problema, a se ne izgubijo v podrobnostih. Ob predstavitvi snovi študentje spoznajo vse pomembnejše dokumente, ki so ravno tako ponekod poenostavljeni:

- **navodila:** za izvajanje celotnega procesa (slika 4) in posameznih delov le-tega,
- **beležke:** porabe časa (slika 5), velikosti izdelkov (slika 6) in napak (slika 7),
- **obrazce:** plan (slika 8), urnike, tedenske in skupne porabe časa in opomnik ter
- **standarde:** za vrste napak, kodiranje in izvajanje pregledov kode

Primeri dokumentov so prikazani na slikah 4-8. Osrednji dokument je plan (slika 8), ki vsebuje celoten prikaz planiranih in izmerjenih podatkov o opravljenem delu.

Namen: postopek razvoja majhnih programov

Vhodni pogoj:
- opis problema
- zbrani zgod. podatki o dejanski porabi časa in velikosti programov
- Plan
- 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 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

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

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

6. Testiranje
- testiraj program
- odpravi vse odkrite napake
- vnesi čas testiranja v Beležko porabe časa

7. Analiza
- do konca izpolni 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 Plan
- vnešeni ustrezni podatki v Beležko porabe časa

Slika 4: Navodilo za izvajanje razvoja majhnih programov

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

Slika 5: Beležka porabe časa

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											

Slika 6: Beležka velikosti izdelkov

Ime: Andrej Novak — Datum: 07.04.2005
Vodja: Pred. Rožanc — 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:						

Slika 7: Beležka napak

PLAN

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	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			

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	

Slika 8: Primer izpolnjenega plana

Na laboratorijskih vajah so študentje PSP preverili tudi v praksi. Uporabljali so ga pri razvoju manjše spletne aplikacije za del poslovanja hotela, katere razvoj je trajal približno 6 tednov. Pri tem so bili organizirani po majhnih skupinah (dva ali trije študenti) in vsaka skupina je razvijala rešitev za svojo nalogo (recimo vselitve/izselitve gostov v sobe ali vzdrževanje podatkov o storitvah hotela). Čeprav je razvoj potekal po skupinah, so študentje PSP uporabljali kot posamezniki.

V času razvoja spletne aplikacije se je vsaka skupina trikrat srečala z asistentom. Na prvem srečanju je skupina prejela nalogo s podrobno obrazložitvijo. Čeprav so bile vse naloge povezane s poslovanjem hotela, smo dovolili skupinam po svoje oblikovati načrt in izgled rešitve. To so skupine predstavile na drugem srečanju, ko so morale pripraviti podatkovni model in opis predvidene funkcionalnosti. Tedaj je vsak posameznik pokazal tudi delno izpolnjen PSP plan s (kakršnimikoli že) ocenami velikosti, porabe časa in števila napak za svoj delež razvoja spletne aplikacije skupine. Na zadnjem srečanju so skupine predstavile izdelano spletno aplikacijo s spremljajočo dokumentacijo, posamezniki pa so oddali izpolnjen plan ter tiste PSP dokumente, ki so jih izpolnili med uporabo PSP-ja.

V skladu s priporočili (Humphrey, 1995) se PSP uvaja postopoma, zato od študentov kot začetnikov nismo pričakovali uporabe PSP-ja v obsegu, kot je bil predstavljen na predavanjih.

Zahtevali smo le spremljanje porabe časa, medtem ko smo jim spremljanje velikosti izdelkov in števila vnešenih in odstranjenih napak priporočali. Na koncu je bilo treba obvezno oddati le (formalno pravilno) izpolnjen plan, pričakovali pa smo vsaj še beležko porabe časa, če že beležk velikosti izdelkov ali odkritih napak ne.

V 3. letnik VSP študija Računalništvo in informatika smeri Programska oprema je letos vpisanih 45 študentov, od katerih je 34 (76 %) obveznosti pri laboratorijskih vajah uspešno opravila. Tako število nam dopušča, da na podlagi vsebine oddanih PSP dokumentov analiziramo uspešnost učenja PSP-ja pri predmetu RPS II. Pri vseh nadaljnih analizah tako velja, da upoštevamo samo študente, ki so opravili obveznosti pri vajah.

5. Analiza uspešnosti učenja PSP-ja

Zbirka oddanih PSP dokumentov neposredno kaže na to, v kolikšnem obsegu so se študentje naučili PSP-ja, posredno pa lahko ugotovimo tudi, če so zares začutili uporabnost PSP-ja.

a) Prvo merilo obsega uporabe je število in vrsta dokumentov, ki so jih oddali študentje, ki so uspešno opravili vaje. Ker imajo dokumenti veliko postavk, smo preverili tudi, če so dokumenti ustrezno izpolnjeni (recimo vse potrebne postavke plana). Rezultati so razvidni iz tabele 1:

PSP dokument:	**Odstotek študentov, ki so oddali dokument:**	**Odstotek študentov, ki so oddali ustrezno izpolnjen dokument:**
Plan	93 %	80 %
Beležka porabe časa	87 %	87 %
Tedenska poraba časa	59 %	14 %
Beležka velikosti	24 %	24 %
Beležka napak	7 %	7 %

Tabela 1: Deleži študentov, ki so oddali posamezne PSP dokumente

Ugotovitev: Glavnina študentov je oddala dokumente, ki so bili zahtevani ali očitno pričakovani glede na to, da smo od njih zahtevali spremljanje časa. To potrjuje tezo, da je na začetku uporabe PSP-ja že spremljanje časa dovolj zahtevno opravilo.

b) Drugo merilo je skladnost PSP dokumentov, ki prikazujejo iste vidike - recimo skladnost beležke porabe časa s skupno porabo časa v planu. S tem preverimo, če so bili podatki zares izmerjeni ali so izmišljeni. To lahko presodimo tudi iz smiselnosti posameznih vrednosti, ki so zapisane v dokumentih (recimo primerno dolgega časa razvoja glede na velikost). Tabela 2 prikazuje ta razmerja za vse tri vidike:

Vidik:	**Skladnost dokumentov:**	**Odstotek študentov, ki so oddali oba dokumenta:**	**Odstotek študentov, kjer so dokumenti skladni:**	**Odstotek študentov, kjer so vrednosti smiselne:**
Čas	Beležka časa : plan	87 %	59 %	52 %
Čas	Beležka časa : ted. poraba časa	45 %	45 %	45 %
Čas	Plan : ted. poraba časa	59 %	59 %	52 %
Velikost	Beležka velikosti : plan	24 %	14 %	14 %
Napake	Beležka napak : plan	7 %	7 %	7 %

Tabela 2: Skladnost in smiselnost oddanih PSP dokumentov

Ugotovitev: Po pričakovanjih je spremljanje časa tisti del, ki je pretežno skladen oziroma ima smiselne vrednosti, vendar ne več tako prepričljivo, kot smo menili doslej. Po podatkih sodeč je realnih dobra polovica podatkov. Podobno bi verjetno lahko sklepali tudi za velikost, ker je merjenje le-te razumljivo, vendar tega zaradi majhnega števila oddanih beležk velikosti ne moremo preveriti. Napake je ustrezno spremljalo samo nekaj (najboljših) študentov, kar ne preseneča – PSP uvaja popolno obvladovanje napak šele v višjih verzijah PSP-ja, ko že imamo veliko izkušenj z vsem ostalim.

c) Mogoča je tudi vsebinska analiza vrednosti posameznih postavk plana (slika 8). Ti podatki so zbrani v tabeli 3, kjer so zanimiva predvsem razmerja med planiranimi in izmerjenimi vrednostmi ter najmanjšimi in največjimi izmerjenimi vrednostmi, medtem ko nam same vrednosti za splošno analizo pravzaprav ne povedo veliko. Več povedo o znanju in izkušnjah posameznika v primerjavi z ostalimi študenti, saj so vsi imeli približno enako obsežne naloge.

Postavka v planu:	**Planirana / izmerjena vrednost:**	**Najmanjša vrednost:**	**Povprečna vrednost:**	**Največja vrednost:**
Čas za izdelavo vrstice programske kode [v min]	Planirana	0.7	3.27	7
	Izmerjena	0.68	2.95	7.3
Velikost končnega izdelka [v vrsticah kode]	Planirana	100	357	600
	Izmerjena	123	434	805
Skupna poraba časa [v min]	Planirana	500	959	1800
	Izmerjena	315	1058	2610
Delež kodiranja v celotnem razvoju [%]	Planiran	30	52	80
	Izmerjen	45	60	72
Skupno število napak	Planirano	28	87	414
	Izmerjeno	36	132	510
Delež napak vnešenih med kodiranjem [%]	Planiran	60	87	100
	Izmerjen	85	97	100
Delež napak odpravljenih med prevajanjem [%]	Planiran	30	66	80
	Izmerjen	52	61	87

Tabela 3: Najmanjše, povprečne in največje vrednosti postavk plana

Ugotovitev: Planirane in izmerjene vrednosti so pri vseh postavkah relativno usklajene, kar kaže na to, da so študenti ob vnosu izmerjenih vrednosti zelo verjetno popravili tudi planirane ocene, česar formalno ne bi smeli. Vendar je to razumljivo, saj so prvo oceno podali brez kakršnihkoli izkušenj z razvojem podobnih izdelkov. Več nam povedo razmerja med najmanjšimi in največjimi izmerjenimi vrednostmi, ki so pri nekaterih postavkah precej velika in kažejo na različno raven znanja in izkušenj posameznikov pri razvoju spletnih aplikacij. Pri spremljanju napak bi lahko govorili tudi o nezanesljivih meritvah.

Povprečne vrednosti nekaterih postavk bi bile bolj smiselne, če bi izločili podatke tistih posameznikov, ki največ odstopajo. To je tudi vidik, ki ga tovrstna analiza predvsem omogoča: iskanje posameznikov, ki v dobrem ali slabem smislu najbolj izstopajo. Rezultate takih posameznikov je treba presojati posebej.

Naše končne ugotovitve o uspešnosti učenja PSP-ja so naslednje:

- študentje so PSP spoznali na primeren način in ga uspešno preverili v praksi;

- večji del študentov je spoznal osnovne principe in začutil njegovo uporabnost, kar še zlasti velja za študente z več programerskimi izkušnjami;
- PSP treba uvajati postopoma in na začetku študentje zares dobro obvladajo le spremljanje časa in (deloma) velikosti;
- študentje imajo različna predznanja in izkušnje; to ni toliko razvidno iz poznavanja principov PSP-ja kot iz njegove uporabe v praksi;
- PSP zahteva temeljito in natančno izvajanje preprostih aktivnosti, kar v praksi ni preprosto.

6. Anketa med študenti

Za celovito analizo potrebujemo še mnenje študentov o vsebini in načinu učenja PSP-ja. Posredno smo mnenja slišali ob različnih stikih s študenti, vendar smo želeli raziskavo opraviti na sistematičen način. Objektivno mnenje študentov o poučevanju PSP-ja smo zato dobili s pomočjo anonimne ankete, ki smo jo med študente razdelili po zaključku predavanj in vaj.

Anketo smo namenili vsem študentom, ki so letos obiskovali predavanja in vaje in nanjo je odgovorilo 76 % študentov – ravno toliko, kolikor jih je opravilo vse obveznosti pri laboratorijskih vajah. Obsegala je 4 strani oziroma 27 vprašanj, od katerih so imela nekatera več (tudi do 8) podvprašanj. Večina vprašanj je kot odgovor pričakovala oceno med 1 in 10, kjer je 1 pomenila najslabši in 10 najboljši odgovor (recimo 1 »nimam nobenih izkušenj«, 10 »imam zelo veliko izkušenj«). Pri ostalih vprašanjih je bil odgovor opisen. Za izpolnjevanje ankete je študent potreboval približno 15 minut. Anketa je še vedno dostopna na spletnem naslovu: http://ltpo.fri.uni-lj.si/predmeti/rps2/ . Anketo so sestavljale tri skupine vprašanj:

a) S prvo skupino vprašanj smo želeli ugotoviti predznanje in izkušnje študenta, s čemer bi zagotovili večjo objektivnost oziroma ustreznejšo interpretacijo odgovorov. Najpomembnejši odgovori te skupine so predstavljeni v tabeli 4.

Vprašanje:	**Odgovor:**
1. Čas študija na fakulteti	18 % več kot 5 let, ostali redno
2. Ocena svojega znanja /izkušenj z razvojem PO	6.21 na lestvici 1-10
3. Ocena pomembnosti področja razvoja PO	7.90 na lestvici 1-10
4. Opis svojih dosedanjih programskih izdelkov	18 % brez izkušenj, ostali 6.64 (1-10)
5. Izkušnje z razvojem PO izven fakultete	50 % nima izkušenj, ostali povprečno 3 leta v skupini z 6.1 članom (eden v skupini 40-150)
6. Izkušnje z zajemom in analizo zahtev	5.16 (1-10), 82 % brez izkušenj, ostali 4.75 let
7. Izkušnje z načrtovanjem in specifikacijo PO	5.27 (1-10), 86 % brez izkušenj, ostali 5 let
8. Izkušnje s kodiranjem in testiranjem	6.93 (1-10), 50 % brez izkušenj, ostali 4.33 let, poznajo 4.68 p.jezike, pov. največji izdelek 25K

9. Izkušnje z nameščanjem in vzdrževanjem PO	6.66 (1-10), 59 % brez izkušenj, ostali 5.29 let
10. Izkušnje s standardnimi pristopi, metodami in tehnikami za razvoj PO	5.20 (1-10), 73 % brez izkušenj, ostali 3.5 let
11. Izkušnje z uporabo orodij za razvoj PO	6.21 (1-10), 55 % brez izkušenj, ostali 4 let, v povprečju poznajo 4.75 orodij
12. Izkušnje s projektnim delom	4.81 (1-10), 23 % brez izkušenj, ostali v skupini s 4.47 člani, projekt traja 4.38 meseca

Tabela 4: Predznanje in izkušnje anketiranih študentov

Ugotovitve: Čeprav so merila študentov različna, lahko iz povprečnih ocen s primerjavo ugotovimo pomembne značilnosti. Najpomembnejša ugotovitev je, da ima kar polovica študentov večletne izkušnje z razvojem PO izven fakultete, čeprav je le 18 % študentov, ki s študijskimi obveznostmi veliko zamujajo. Svoje poznavanje področja razvoja PO študentje ocenjujejo nadpovprečno in menijo, da je zelo pomembno za njihovo poklicno pot. To dokazujejo tudi njihove ocene poznavanja posameznih področij razvoja PO, čeprav delež študentov z izkušnjami na posameznih področjih daje realnejšo sliko. Po pričakovanjih anketiranci najbolj obvladajo kodiranje in testiranje, temu sledi vzdrževanje in nameščanje. Dobro ocenjujejo še svoje poznavanje orodij, medtem so ostala področja ocenjena povprečno. Z vidika PSP-ja je zanimiv še rezultat vprašanju 12; čeprav ima velika večina izkušnje s projektnim delom, je ocena svojega znanja na tem področju najnižja. Verjetno se študenti na tem področju bolj zavedajo težav in ne dajejo nevtralnih ocen (5), kot so jih pri področjih, ki jih niso zares poznali.

b) Druga skupina je zajemala vsebinska vprašanja o vseh tematikah, ki so jih študentje spoznali na predavanjih in vajah. Poudarek je bil predvsem na primerjavi PSP-ja z drugimi tematikami. Pomembnejši odgovori se nahajajo v tabeli 5.

Vprašanje:	**Odgovor:**
1. Izbira snovi predmeta	8.07 na lestvici 1-10
2. Obsežnost in zahtevnost snovi	7.52 na lestvici 1-10
3. Ocena tematike »CMM in ostali modeli«	pomembnost 6.45, predstavitev 8.35, primerna in zahtevna
4. Ocena tematike »TSP«	pomembnost 7.21, predstavitev 8.35, primerna in uporabna
5. Ocena tematike »PSP«	pomembnost 7.65, predstavitev 8.75,

	zanimiva, primerna in zelo uporabna
6. Ocena tematike »SSADM«	pomembnost 7.07, predstavitev 8.68, zahtevna, uporabna
7. Ocena tematike »Scrum in XP«	pomembnost 6.88, predstavitev 8.63, zanimiva in primerna
8. Ocena tematike »PL-SQL«	pomembnost 7.13, predstavitev 8.70, zelo primerna in zelo uporabna
9. Ocena tematike »Oraclova orodja«	pomembnost 6.55, predstavitev 8.60, uporabna in zahtevna
10. Ocena tematike »Spletne tehnologije«	pomembnost 7.14, predstavitev 7.74, zelo uporabna in zanimiva

Tabela 5: Vsebinska vprašanja in odgovori v anketi

Ugotovitve: Ocene študentov so za vsa področja dobre, še zlasti pa to velja za predstavitev snovi. Anketiranci so zadovoljni z izbrano tematiko predavanj in vaj, pa tudi obseg in zahtevnost se jim zdita še kar primerna. Po potrebnosti so med tematikami predavanj na prvo mesto postavili PSP in s tem potrdili naša pričakovanja, čeprav je na to gotovo nekoliko vplival tudi časovni obseg predavanj iz te snovi. Če pogledamo mnenja tistih anketirancev, ki že imajo izkušnje na področju razvoja PO, postane slika še bolj jasna; saj so ravno ti tisti, ki so PSP-ju pripisali najboljše ocene.

Študentje so visoko ocenili tudi TSP, medtem ko je predstavitev modelov dobila precej nižjo oceno. To pomeni, da smo se odločili prav, ko smo se odločili za poučevanje kakovosti PO na bolj neposreden način. Večina anketirancev tudi meni, da je PSP v primerjavi z drugimi področji zanimiv in zelo uporaben, vendar so tudi zahtevnost uporabe ocenili kot primerno. To je treba tolmačiti v primerjavi z drugimi tematikami (recimo CMM-jem), ki so bolj abstraktne in za študente težje predstavljive. V prejšnjem poglavju je analiza PSP dokumentov pokazala nekoliko drugačno sliko. Ostale tematike so se uvrstile po pričakovanjih: študentom so bližje orodja in tehnike, ki so jih lahko konkretno preizkusili na vajah, zato so te ocenili bolje.

c) Tretja skupina vprašanj je bila namenjena študentskim mnenjem o kakovosti izvajanja predavanj in vaj. Pri tem nismo spraševali po določeni tematiki, temveč za predmet v celoti. Poleg ocen so tu študenti pripisali še svoje splošne pripombe. Rezultati so v tabeli 6.

Vprašanje:	**Odgovor:**

1. Ocena razumljivosti predavanj in vaj	8.97 na lestvici 1-10
2. Ocena zanimivosti predstavitve predavanj	7.35 na lestvici 1-10
3. Ocena navajanja k razmišljanju	7.70 na lestvici 1-10
4. Ocena obsega predstavljene snovi	8.34 na lestvici 1-10
5. Ocena literature in spletne strani	8.74 na lestvici 1-10
6. Ocena dostopnosti predavatelja in asistenta	9.74 na lestvici 1-10
7. Ocena izvajanja avditornih vaj	8.88 na lestvici 1-10
8. Ocena seminarskih nalog	9.43 na lestvici 1-10
9. Ocena pisnih in ustnih izpitov	8.26 na lestvici 1-10
10. Ocena odnosa predavatelja in asitenta do študentov	9.80 na lestvici 1-10
11. Splošne pripombe	Več praktičnih primerov

Tabela 6: Mnenjska vprašanja in odgovori v anketi

Ugotovitve: Glede izvedbe predavanj in vaj pri predmetu RPS II ni resnejših pripomb, saj so vse ocene razen dveh v območju med 8 in 10. Kritika študentov meri le na zanimivost predstavitve in premajhno navajanje k razmišljanju, kar je posledica velikega obsega celotne snovi predmeta in načina podajanja snovi s projekcijo vnaprej pripravljenih prosojnic. Pri nekaterih tematikah (recimo CMM-ju) je bila snov tako zares podana relativno hitro in brez veliko praktičnih primerov, kar pa za področje PSP-ja nikakor ni bilo res.

7. Zaključek

Področje kakovosti PO je nedvomno področje, ki mu je treba pri poučevanju računalniških vsebin posvetiti več pozornosti. Ker so za slovenski trg programske opreme zanimivi predvsem pristopi, ki so primerni za manjša računalniška podjetja in posameznike, je bil poudarek na modelu PSP za poučevanje kakovosti PO primerna odločitev.

To potrjuje tudi analiza PSP dokumentov in pravzaprav nasploh uspešnost izpitnih rezultatov pri predmetu RPS II. Pokazalo se je, da so študentje uspešno pridobili znanja iz PSP-ja in so PSP začeli tudi praktično uporabljati, čeprav bodo za učinkovito uporabo - glede na zahtevnost in potrebno postopnost uvajanja PSP principov - potrebovali še veliko truda. Tudi mnenjska anketa in neformalni pogovori s študenti kažejo, da je model zanimiv in ga študentje visoko cenijo. V primerjavi z drugimi tematikami predmeta RPS II so mu prisodili najvišjo oceno, še zlasti pa so pohvalili njegovo uporabnost. Ker so PSP-ju najboljše ocene prisodili boljši študentje, ki že imajo izkušnje z razvojem PO, lahko takim rezultatom še bolj verjamemo.

To nas utrjuje v prepričanju, da bomo v naslednjem študijskem letu s poučevanjem PSP-ja nadaljevali. Opravljena analiza nam bo omogočila, da bomo to storili še bolje in učinkoviteje.

Literatura

Amam, K. E., Drouin, J-N., Melo, W., (1998): »SPICE: The Theory and Practice of Software Improvement and Capability Determination«, IEEE Computer Society Press, Los Alamitos.

Beck, K.,(2000): »Extreme Programming Explained: Embrace Change«, Reading, Addison-Wesley.

Feuerstein, S.,(2002): »Oracle PL/SQL Programming, Third Edition«, O'Railly, ISBN: 0-596-00381-1.

Goodland, M., Slater, C., (1995): »SSADM Version 4: A Practical Approach«, McGraw-Hill, London.

Greenwald, R., Milbery, J.,(2001): »Oracle 9i AS Portal Bible«, J. Wiley & Sons Publishing, ISBN 0-7645-4749-6.

Haase, V., Messnarz, R., Koch, G., Kugler, H. J., Dicrinis, P., (1994): »BOOTSTRAP: Fine-tuning process assesment«, IEEE Software, zv. 11, št. 4, str. 25 – 35.

Humphrey W.S., (1995): »The Discipline for Software Engineering«, Addison-Wesley, ISBN 0-201-54610-8.

Humphrey W.S., (1997): »*Introduction to the Personal Software Process«,* Addison-Wesley Publishing Company, Reading (ZDA), ISBN 0-201-47719-X.

Humphrey W.S., (1999): »Pathways to Process Maturity: The Personal Software Process and Team Software Process«, SEI Interactive, zv. 2, št. 2.

Humphrey W.S., (2000a): »Introduction to the Team Software Process«, Addison Wesley, ISBN 0-201-47719-X.

Humphrey W.S., (2000b): »The Personal Software Process (PSP)«, SEI tehnično poročilo CMU/SEI-2000-TR-022.

Humphrey W.S., (2000c): »The Team Software Process (TSP)«, SEI tehnično poročiloCMU/SEI-2000-TR-023.

ISO, (2004a): »ISO 9000:2000, Quality management systems«, standard, International Organization for Standardization.

ISO, (2004b): »ISO/IEC 90003:2000, Software engineering -- Guidelines for the application of ISO 9001:2000 to computer software«, standard, International Organization for Standardization.

Koletzke, P., Dorsey, P., (1999): »Oracle Developer Advanced Forms and Reports«, McGraw-Hill Publishing, ISBN: 0-07-212048-7.

Kuvaja, P. , Simila, J., Krzanik, L.,Bicego, A., Koch, G., Saukonen, S., (1994): »*Software Process Assessment and Improvement: the BOOTSTRAP approach«*, Blackwell Publishers, Oxford.

Paulk M. C. et al, (1993): »Key Practices of the CMM, Version 1.1.«, SEI tehnično poročilo CMU/SEI-93-TR-025.

Schwaber, K.; Beedle, M.,(2001): »Agile Software Development With Scrum«, Upper Saddle River, NJ, Prentice-Hall.

SEI, (2002): »Capability Maturuty Model Integration (CMMI^{SM}), Version 1.1 - CMMI^{SM} for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS,V1.1), Staged Representation«, Tehnično poročilo CMU/SEI-2002-TR-012.

Sommerville, I., (2004): »Software Engineering«, Addison-Wesley Publishing Company, sedma izdaja, New York.

Igor Rožanc *je predavatelj na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Raziskovalno se ukvarja s področjem tehnologije programske opreme, še zlasti z modeli za zagotavljanje kakovosti programske opreme. Sodeluje tudi pri razvoju računalniško podprtih informacijskih sistemov za področje visokega šolstva (projekt e-Študent). Pedagoške izkušnje si že dobro desetletje nabira kot asistent pri predmetih s področja programiranja in tehnologije programske opreme. V zadnjem času na Fakulteti za računalništvo in informatiko predava predmete Osnove programiranja I, Osnove algoritmov in podatkovnih struktur I ter Razvoj programskih sistemov II.*

Viljan Mahnič *je izredni profesor in prodekan za pedagoško delo na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Od leta 2000 je predstojnik Laboratorija za tehnologijo programske opreme, kjer se ukvarja se z s posebnim poudarkom na informacijskih sistemih. Od leta 1996 je predstavnik Slovenije v EUNIS (European University Information Systems Association), od leta 2002 pa tudi član sveta direktorjev omenjene organizacije. Poleg tega aktivno deluje na področju računalniškega izobraževanja: je član Republiške predmetne komisije za predmet Računalništvo in član Programskega sveta za informatizacijo šolstva.*