UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

MATIJA KRAJNC

OCENJEVANJE POTREBNEGA ČASA ZA IZVEDBO DVEH STOPENJ RAZVOJA PROGRAMSKE OPREME

MAGISTRSKO DELO

Mentor: prof. dr. Miran Mihelčič

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št.: 111-MAG-ISO/2010
Datum: 17.06.2010

Matija KRAJNC, univ. dipl. inž. rač. in inf.
Ljubljana

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Ocenjevanje potrebnega časa za izvedbo dveh stopenj razvoja programske opreme**

Estimating the time needed for execution of two phases in software development

Tematika naloge:

Programska oprema oz. informacijska podpora za poslovne procese postaja po svetu vedno bolj pomembna, saj je z njo mogoče delno ali popolnoma podpreti delovanje poljubne združbe in tako doseči njeno bolj učinkovito ter posledično bolj uspešno poslovanje. Sposobnost nosilca za učinkovit, kakovosten, časovno kratek in ekonomsko sprejemljiv razvoj programske opreme postaja zato tekmovalna prednost.

Vsak projekt razvoja programske opreme, od najmanjšega pa vse do največjega, potrebuje za svojo izvedbo ustrezne prvine poslovnega procesa. Uspešna izvedba projekta je pogojena z navzočnostjo ustreznih poslovnih prvin, katerih obseg oz. število je treba pogosto predvideti oziroma oceniti v začetnih stopnjah projekta ali celo prej. Pri razvoju programske opreme je čas sodelujočih, potreben za izvedbo projekta, tista poslovna prvina, za katero ravnatelj projekta najpogosteje poda (ne)ustrezno oceno. Oceno lahko namreč ravnatelj pridobi na podlagi osebnih, subjektivnih izkušenj ali pa na podlagi uporabe ustreznih metod, ki zagotavljajo bolj objektivne in posledično bolj natančne ocene potrebnega časa. Pomembno je, da si ravnatelj projekta ustvari čimbolj natančno oceno časa za posamezne stopnje oziroma aktivnosti na projektu že pred začetkom uresničevanja projekta.

V nalogi predstavite obstoječe načine ocenjevanja potrebnega časa za izdelavo programske opreme. Pojasnite, zakaj je ocenjevanje potrebnega časa pomembno za kakovosten razvoj programske opreme. Opišite najpogosteje uporabljene metode za ocenjevanje časa, še posebej metodo funkcijskih točk (*angl. Function Points Analysis*), metodo točk primerov uporabe (*angl. Use Case Points Method*) in metodo poenostavljenih točk primerov uporabe. Na primeru petih projektov iz gospodarstva preverite učinkovitost metode poenostavljenih točk primerov uporabe za dve stopnji razvoja programske opreme: analiza (oz. razčlenjevanje) in načrtovanje. Pri tem zberite vse informacije, potrebne za pripravo ocene potrebnega časa, in jih ustrezno uporabite pri raziskavi.

Mentor:
prof. dr. Miran Mihelčič

Dekan:
prof. dr. Franc Solina

Zahvala

Zahvaljujem se mentorju prof. dr. Miranu Mihelčiču za ves njegov trud, pomoč in koristne nasvete.

Najlepše se zahvaljujem svoji družini za njihovo vsestransko podporo v času študija ter Jani za potrpežljivost in pomoč.

Hvala tudi prijateljem, ki so na različne načine pomagali pri ustvarjanju tega magistrskega dela.

Kazalo

Kazalo slik

Kazalo preglednic

Kazalo enačb

Kratice

CICS	*angl. Customer Information Control System*
COCOMO	*angl. COnstructive COst MOdel*
EF	*angl. Environmental Factor*
E-R	*angl. Entity-Relationship*
IFPUG	*angl. International Function Points User Group*
LOC	*angl. Lines Of Code*
MMRE	*angl. Mean Magnitude of Relative Error*
MRE	*angl. Magnitude of Relative Error*
OBS	*angl. Organizational Breakdown Structure*
PBS	*angl. Product Breakdown Structure*
PDF	*angl. Portable Document Format*
PPP	prvina poslovnega procesa
PU	primer uporabe
SOMA	*angl. Service-Oriented Modeling and Architecture*
SSDAM	*angl. Structured-Systems Analysis and Design Method*
RUP	*angl. Rational Unified Process*
TCF	*angl. Technical Complexity Factor*
UAW	*angl. Unadjusted Actor Weights*
UCP	*angl. Use Case Points*
UML	*angl. Unified Modelling Language*
UUCP	*angl. Unadjusted Use Case Points*
UUCW	*angl: Unadjusted Use Case Weights*
WBS	*angl. Work Breakdown Structure*
XML	*angl. Extensible Markup Language*

Ocenjevanje potrebnega časa za izvedbo dveh stopenj razvoja programske opreme

POVZETEK

V magistrskem delu preučujem sprejemljivost metode poenostavljenih točk primerov uporabe za oceno potrebnega časa izvedbe stopenj analiziranja in načrtovanja programske opreme. Z uporabo omenjene metode želim bolje napovedati potreben čas za izvedbo navedenih aktivnosti. Namen naloge je na preprost in hiter način omogočiti učinkovitejše ocenjevanje potrebnega časa že ob samem začetku izvajanja projekta razvoja programske opreme za posamezne stopnje (oz. dejavnosti) razvoja, kot sta analiziranje in načrtovanje programske opreme.

Pri ocenjevanju potrebnega časa za stopnji analiziranja in načrtovanja sem uporabil načrt dela za uresničitev projekta oz. WBS (*angl. work breakdown structure*), s katerim sem zaznal in preštel najmanjše zaokrožene enote delovanja programske opreme. Te enote, transakcije oz. informacijsko podprta poslovna opravila, sem v nadaljevanju uporabil kot osnovno sestavino za pripravo ocene potrebnega časa. Nato sem najprej preučil teoretične značilnosti različnih splošnih metod za oceno potrebnega časa, v nadaljevanju pa sem se osredotočil na preučevanje sprejemljivih metod za oceno potrebnega časa pri razvoju programske opreme. V okviru analize sem zaznal tri metode, ki bi jih lahko uporabil. Te metode so: metoda funkcijskih točk, metoda točk primerov uporabe in metoda poenostavljenih točk primerov uporabe. Zaradi zmožnosti ocenjevanja ob samem začetku uresničevanja projekta, preprostosti uporabe in ker je mogoče z metodo ocenjevati čas za posamezne stopnje (oz. dejavnosti) razvoja. V raziskavi sem uporabil metodo poenostavljenih točk primerov uporabe.

Poglavitni učinek magistrskega dela je ugotovitev, da je mogoče z metodo poenostavljenih točk primerov uporabe pripraviti sprejemljive ocene potrebnega časa za stopnji analiziranja in načrtovanja programske opreme bolje kot z metodo funkcijskih točk in metodo točk primerov uporabe. Napaka ocen za stopnji analiziranja in načrtovanja pri večini projektov ne presega vrednosti 25 odstotkov dejansko porabljenega časa. Dodaten učinek dela je ugotovitev, da ima na kakovost ocene večji vpliv število uresničenih projektov kot pa podobnost med uresničenimi projekti, katerih podatke o dejansko porabljenem času sem uporabil v raziskavi. Vpliv dejavnika podobnosti med uresničenimi projekti na pripravo ocene potrebnega časa bi bilo nedvomno priporočljivo podrobneje raziskati v prihodnje.

Raziskava je bila izvedena v obdobju treh let, in sicer na petih uspešno uresničenih projektih. Izkazalo se je, da smo na zadnjem, petem projektu pridobili sprejemljive ocene potrebnega časa za stopnji analiziranja in načrtovanja programske opreme, ki jih je ravnatelj projekta lahko uporabil za bolj kakovostno in učinkovito uresničitev naslednjega, šestega projekta. Uporabljena metoda se je izkazala kot ustrezna.

Ključne besede: ocenjevanje potrebnega časa, primer uporabe, razvoj programske opreme, analiziranje, načrtovanje.

ABSTRACT

The following Masters thesis studies the acceptance of simplified use case point method for time (effort) estimation of execution, analysis and design phase in software development. With this method I wanted to estimate the required time for execution of a particular phase. The aim of this thesis is to provide a simple and fast way for efficient time estimation at the beginning of software development project for particular development phases, like the analysis and design phase.

WBS (Work Breakdown Structure) was used as a blueprint for implementation of the project and as a basis for estimation of the required time for analysis and design phase. With WBS the smallest units of work which need to be implemented in the software were identified and counted. These units of work were used as a basic input for time (effort) estimation. In the next step theoretical background for different, generally used methods for time (effort) estimation were analyzed and at the end acceptable methods for effort estimation in software development projects were analyzed in greater detail. Three acceptable methods were identified within analysis: function point method, use case point method and simplified use case point method. The simplified use case point method was proposed and used in this research because of the following features: the ability to estimate effort in the earlier phases of project, the simplicity of use and the ability to estimate the effort needed for particular phase of software development project.

The main result of my thesis is a confirmation that the simplified use case point method can provide acceptable effort estimation for analysis and design phase in software development projects. In the most projects used in this research the difference (error) between effort estimation and actual effort for analysis and design phase was smaller than 25 percent. Additional result of my thesis is also a finding that the quality of effort estimation with the simplified use case method more depends on the number of used projects (or projects history) than the similarity between the used projects in this research. It's recommended that the influence of the similarity factor is to be researched in greater detail in the future.

The research was carried out in the period of three years on five successfully executed software projects. In the last, the fifth project the method provided acceptable effort estimation for analysis and design phase. These estimations were then used by the project manager for quality and efficient project implementation. The used method has been proven as acceptable.

Keywords: effort estimation, time estimation, use case, software development, analysis, design.

1 Uvod

1.1 Izziv

Ena izmed najtežjih nalog ravnatelja katerekoli vrste projekta je podati ocene količine potrebnih prvin za uspešno izvedbo projekta. Za naročnika projekta je pogosto najpomembnejša prvina čas, potreben za uspešno izvedbo projekta. Uspešen zaključek projekta pred rokom ponavadi ustrezno nagradijo, prekoračitev časovnega okvirja pa pogosto kaznujejo. Sposobnost za učinkovito, kakovostno, časovno in ekonomsko sprejemljivo uresničitev projekta (razvoja programske opreme) postaja zato tekmovalna prednost. Ključen dejavnik za dosego tekmovalne prednosti predstavlja kakovosten projektni načrt in učinkovito uresničevanje slednjega ([22] Kusumoto in ostali, 2004; 292). Zelo pomembno je, da že pred uresničevanjem projekta poskušamo napovedati in oceniti (ne)želene dogodke. Ko govorimo o napovedovanju, imamo v mislih naslednje pojme: velikost, vložek napora (dela), čas razvoja, uporabljena tehnologija in kakovost. Najpomembnejša je ocena velikosti oz. obseg vsebine dela, ki jo bomo informacijsko podprli, posledično pa ocena potrebnega časa za izvedbo tega dela.

V preteklosti je bilo opravljenih že veliko raziskav na temo uspešnosti ocenjevanja potrebnega dela oz. časa za razvoj programske opreme. Raziskave kažejo, da so ocene ponavadi preveč optimistične in da so ocenjevalci preveč prepričani v pravilnost svojih ocen ([44] Wikipedia, Software development effort estimation, 2009). Kot primer prevelike prepričanosti raziskovalci navajajo podatek, da je v povprečju strokovnjak s področja razvoja programske opreme v 90 odstotkih oz. skoraj popolnoma prepričan, da je njegova ocena ustrezna. Zaključek projekta ponavadi pokaže, da bi morala biti raven njegove, skoraj popolne prepričanosti nižja za 20 do 30 odstotkov. Poročilo združenja Standish group ([45] Standish group's CHAOS Report, 2009) kaže na to, da 32 odstotkov projektov uspešno zaključijo v predvidenem časovnem in finančnem obsegu, in sicer z zahtevano kakovostjo in obsegom, 44 odstotkov končajo s prekoračitvijo časovnega ali finančnega obsega, in sicer z zmanjšanim obsegom ali kakovostjo, 24 odstotkov projektov pa zaključijo neuspešno (so predčasno prekinjeni ali pa se njihov izložek ne uporablja).

Pri razvoju programske opreme danes največ uporabljajo metodo COCOMO, metodo ocenjevanja na podlagi mnenja strokovnjaka in metode, ki temeljijo na pristopu ocenjevanja časa na podlagi obsega programske opreme. Danes sta tipični predstavnici tega pristopa metoda funkcijskih točk (*angl. function points*) in metoda točk primerov uporabe (*angl. use case points*). Praktične izkušnje kažejo, da je pomanjkljivost metode COCOMO ta, da ocenjevanje števila vrstic kode nehote poudarja zgolj stopnjo izvedbe (oz. programiranja) programske opreme. Za razliko od metode COCOMO ostale metode, navedene v tem poglavju, zaobsegajo tudi oceno za ostale stopnje razvoja programske opreme. Kljub široki uporabi omenjenih metod pa imajo predhodno opisane metode še vedno dve pomanjkljivosti. Prva je ta, da je z metodami mogoče pripraviti ocene potrebnega časa za projekt dokaj pozno, ponavadi po opravljeni stopnji analiziranja programske opreme. Druga pa je ta, da z omenjenimi metodami ni mogoče pripraviti ocene potrebnega časa za nekatere stopnje razvoja programske opreme, kot sta stopnji analiziranja in načrtovanja.

Prvo pomanjkljivost rešuje metoda poenostavljenih točk primerov uporabe, ki sta jo postavila Robiola in Orosco ([35] 2008). Avtorja sta si pri iskanju nove metode zadala nalogo, da poiščeta preprostejšo in natančnejšo metodo, kot pa sta metoda funkcijskih točk in metoda

točk primerov uporabe. Z novo metodo sta želela doseči, da ocenjevanje potrebnega časa lahko izvedemo dovolj zgodaj na projektu in da zmanjšamo velikost napake ocene potrebnega časa. Novo metodo sta osnovala na primerih uporabe, vendar na nekoliko prilagojen način, saj sta velikost primerov uporabe izmerila s samo dvema neodvisnima sestavinama: transakcijo in entiteto. Uporabo slednjih pojmov avtorja utemeljita z dejstvom, da lahko vsako programsko opremo opazujemo z dveh vidikov: z vidika delovanja (oz. transakcij) in z vidika podatkov. V opravljeni raziskavi sta avtorja potrdila ustreznost uporabe metode z vidika delovanja (oz. transakcij), z vidika podatkov (oz. entitet) pa metoda ni ponudila želenih rezultatov. Z metodo poenostavljenih točk primerov uporabe v okviru tega dela želim odpraviti tudi drugo pomanjkljivost. Z njo sem se srečal v času redne zaposlitve pri prejšnjem delodajalcu, kjer sem bil v okviru uresničevanja petih projektov zadolžen za opravljanje delovnih nalog iz stopenj analiziranja in načrtovanja programske opreme.

1.2 Namen in cilji

Namen naloge je omogočiti učinkovitejše ocenjevanje potrebnega časa ob samem začetku izvajanja projekta razvoja programske opreme in to za posamezne stopnje (oz. dejavnosti) razvoja, kot sta analiziranje in načrtovanje programske opreme.

Za uresničitev tega namena sem si v nalogi določil tri cilje. Prvi cilj je poiskati ustrezno metodo za ocenjevanje potrebnega časa, ki omogoča boljšo pripravo ocene ob samem začetku izvajanja projekta razvoja programske opreme. V okviru drugega cilja je potrebno izbrati ustrezen nabor projektov, na katerih bo potekala raziskava naloge. Zadnji, tretji cilj, je izvesti raziskavo, ki bo potrdila ustrezno natančnost ocen potrebnega časa, pridobljenih na podlagi izbrane metode. Ocene potrebnega časa bom pripravil za izvedbo stopenj analiziranja in načrtovanja programske opreme.

1.3 Metode dela

Na osnovi dostopne literature bom najprej preučili metode za ocenjevanje potrebnega časa v proizvajalni in storitveni dejavnosti. Preučil bom možnosti učinkovitega popisa zgradbe poslovnega učinka (storitve ali proizvoda), ki bo služil kot osnova za pripravo ocene potrebnega časa. Nato bom preučil metode ocenjevanja časa pri razvoju programske opreme, še posebej metode, ki omogočajo pripravo ocene ob samem začetku uresničevanja projekta in to oceno za posamezne stopnje razvoja programske opreme. Podrobneje bom preučil metodo poenostavljenih točk primerov uporabe. Nato bom izbral nabor petih projektov, na katerih bom izvedel raziskavo, ki bo potrdila ali ovrgla ustrezno natančnost ocen potrebnega časa, pridobljenih na podlagi metode poenostavljenih točk primerov uporabe. Ocene potrebnega časa bom pripravil za uresničitev stopenj analiziranja in načrtovanja programske opreme. Ob izvajanju vseh petih projektov bom zagotovil dosledno in nepristransko zbiranje ter analizo podatkov o dejansko porabljenem času za izvedbo stopenj analiziranja in načrtovanja programske opreme. Ocena potrebnega časa bo sprejemljivo natančna takrat, ko napaka pri vsaj 75 odstotkih projektov ne bo presegla 25 odstotkov dejansko porabljenega časa.

1.4 Zgradba naloge

Magistrsko nalogo bom vsebinsko razdelil na dva večja sklopa. V prvem bom najprej predstavil teoretične osnove ocenjevanje časa nasploh. Sledil bo pregled sestave WBS, ki služi kot osnovno orodje za urejen opis zgradbe poslovnega učinka oz. izida projekta. Preučil bom možnosti različnih načinov priprave sestave WBS in dodatne načine opisa zgradbe poslovnega učinka. Nato bom opredelil osnovne pojme, ki jih srečujemo pri ocenjevanju potrebnega časa, in predstavil najbolj pogoste splošne metode za ocenjevanje časa. Nadaljeval bom z razčlenitvijo trenutnega stanja na področju ocenjevanja časa pri razvoju programske opreme. Preučil bom izhodišča in različne metode, ki jih uporabljamo pri razvoju programske opreme. Nekoliko podrobneje bom predstavil prednosti in slabosti metode funkcijskih točk ter metode točk primerov uporabe. Na koncu prvega dela bom poglobljeno predstavil metode poenostavljenih točk primerov uporabe ter navedel razloge za izbiro te metode v okviru tega dela. Podrobneje bom predstavil pomen zgodovinskih podatkov pri pripravi ocene, čas

priprave ocene potrebnega časa in možnost priprave ocene za posamezno stopnjo razvoja programske opreme.

V drugem delu bom opisal izvedbo raziskave, v kateri sem uporabil metodo poenostavljenih točk primerov uporabe za pripravo ocene potrebnega časa za stopnji analiziranja in načrtovanja na petih različnih projektih razvoja programske opreme. Ocene potrebnega časa bom pripravil na podlagi treh pristopov: na podlagi podatkov vseh uresničenih projektov, na podlagi podatkov že uresničenih podobnih projektov in na podlagi podatkov sproti uresničenih projektov. Na podlagi nepristranske analize pridobljenih podatkov bom odgovoril na vprašanje, ali metoda poenostavljenih točk primerov uporabe ponuja sprejemljive ocene potrebnega časa za stopnji analiziranja in načrtovanja programske opreme. Nalogo bom končal s predstavitvijo sklepnih ugotovitev in napotkov za bodoče raziskovalce.

2 Ocenjevanje časa

2.1 Osnove

Vsak projekt, od najmanjšega pa vse do največjega, potrebuje za svojo izvedbo ustrezne prvine poslovnega proces (v nadaljevanju PPP). Nekateri ljudje verjamejo, da je osnovni dejavnik uspešnosti projekta natančnost ocene potrebnih prvin, kot so ocene potrebnega časa, ljudi, denarja ali ostalih prvin, potrebnih za izvedbo projekta ([19] Kerzner, 2006; 247). Za naročnika projekta je pogosto najpomembnejši dejavnik čas oziroma datum, do katerega naj bi bil projekt uspešno izveden. Pogosto se dogaja, da se prekoračitev časovnega okvirja projekta kaznuje s pogodbeno določenimi kaznimi. In tudi obratno: uspešen zaključek projekta pred rokom ustrezno nagradijo ([9] Fitzsimmons, 1998; 212). V okviru te naloge bom obravnaval predvsem oceno potrebnega časa za izvedbo projekta ali le dela projekta. Ocena časa je količinska enota, ki pove, koliko enot časa je potrebnih za izvedbo (določenega dela) projekta ali neke dejavnosti znotraj projekta. Časovne enote so ponavadi podane v urah ali dnevih, pri velikih projektih pa lahko srečamo ocene, podane v tednih ali mesecih. Te ocene so nato osnova za uspešno načrtovanje izvedbe projekta ([10] Heldman in drugi, 2005; 257). Po Heldmanu in drugih ([10] 2005; 254) je za pripravo ocen in za načrtovanje izvedbe projekta potrebno toliko časa kot za samo izvedbo in kontrolo projekta.

Ocenjevanje časa za izvedbo projekta se začne že v začetnih stopnjah projekta. Seveda so te ocene zelo grobe in dokaj nenatančne. Z razvojem projekta in s pridobivanjem vse podrobnejših in natančnejših informacij o projektu se ocene izboljšujejo in postajajo vse bolj natančne. Vir informacij o projektu so lahko sprotni podatki o konkretnem projektu, predhodne izkušnje ljudi, ki delajo na projektu, ali pa zgodovinski podatki o že izvedenih projektih ([10] Heldman in drugi, 2005; 259).

Nekaj virov informacij za pripravo ocene potrebnega časa je navedenih že v prejšnjem poglavju. K njim lahko prištejemo še: dejavnike okolja, značilnosti posameznih dejavnosti oziroma aktivnosti projekta, značilnosti samega podjetja itd. Po mnenju mnogih avtorjev pa je najpomembnejši vir informacij urejena zgradba vseh sestavnih delov projekta ali poslovni učinek projekta (proizvod ali storitev). Tako je lahko sestavni del posamezna dejavnost ali aktivnost, opravilo, sestavni del proizvoda ... Ti sestavni deli pa so nato urejeni v zgradbo, ponazorjeno bodisi v obliki podrobnega načrta proizvoda (*angl. product breakdown structure* oz. krajše PBS), bodisi podrobnega načrta dela za izvedbo projekta (*angl. work breakdown structure* oz. krajše WBS) bodisi podrobnega načrta izvedbe posamezne storitve (*angl. service blueprint*) itd. Nekateri avtorji ([7] Devaux, 1999; 74) omenjajo tudi podrobne načrte organizacijskih sestav (*angl. organizational breakdown structure* oz. krajše OBS) in podrobne načrte sestave poslovnih funkcij ([7] Devaux, 1999; 78). Pogosto je sestava prikazana na slikoven način. Zanimivo je, da se v literaturi za opis urejene zgradbe vseh sestavnih delov projekta najpogosteje omenja podroben načrt dela za izvedbo projekta (v nadaljevanju WBS), ne glede na to, ali gre za storitveno ali za proizvajalno dejavnost. Kaj več bomo o tem lahko prebrali v nadaljnih poglavjih.

Na tem mestu je potrebno spregovoriti tudi nekaj besed o opredelitvi osnovnih sestavnih delov projekta. Nekaj vrst sestavnih delov je že bilo omenjenih v prejšnjem odstavku. Ti so lahko: dejavnost ali aktivnost, opravilo, sestavina poslovnega učinka, transakcija ... V literaturi na to temo srečamo kar nekaj različnih opredelitev sestavnih delov, kar povzroča veliko nesporazumov. Tako npr. Fitzsimmons ([9] 1998; 87) piše, da je storitev sestavljena iz

medsebojno povezanih transakcij, v naslednjem odstavku pa že omenja aktivnosti kot sestavni del storitve. V nadaljevanju ni navedena opredelitev za razlikovanje med transakcijo in aktivnostjo. Na drugi strani pa Hope in Mühlemann ([14] 1997; 231) omenjata, da je storitev sestavljena iz posameznih operacij, v nadaljevanju pa omenjata tako aktivnost ([14] 1997; 234) kot tudi opravilo ([14] 1997; 255) kot sestavni del storitve. V nadaljevanju ponovno ni navedena nobena opredelitev, ki bi opisovala razliko med operacijo, opravilom in aktivnostjo. V poglavju o storitveni dejavnosti bomo poskusili razčistiti te nesporazume in pripraviti enotno opredelitev sestavnih delov storitve. Slednje predstavlja osnovo za nadaljnje delo in izvedbo raziskave o možnostih ocenjevanja časa na projektu razvoja programske opreme.

Napisati je potrebno tudi nekaj besed o samih metodah za pripravo ocene potrebnega časa. Po Hopu in Mühlemannu ([14] 1997; 255) jih v osnovi delimo na:

- neposredne in
- posredne.

Neposredne metode so tiste, pri katerih naredijo oceno na podlagi neposrednih meritev časa, potrebnega za izvedbo določenega opravila ali aktivnosti. Med te metode spadajo: metoda opazovanja časa (*angl. time study*), metoda samostojnega beleženja časa (*angl. self-logging*), metoda vzorčenja aktivnosti (*angl. activity sampling*) itd. Med posredne metode pa spadajo: metoda prednastavljenega časa gibanja (*angl. predetermintation motion time system*), metoda združitve v celoto (*angl. synthesis*), metoda ocene strokovnjaka (*angl. expert estimation*) itd. Več o posameznih metodah bo sledilo v nadaljevanju.

2.2 Predstavitev WBS (*angl. work breakdown structure*)

2.2.1 Uvod

V predhodnem poglavju je zapisano, da je po mnenju mnogih avtorjev najpomembnejši vir informacij za dobro oceno časa urejena zgradba vseh sestavnih delov projekta ali poslovnega učinka (proizvoda ali storitve). Sestavni del projekta je lahko enota dela, ki je potrebna za izvedbo določenega dela projekta, lahko je polproizvod, ki ga je potrebno vgraditi v končni proizvod, lahko je del storitve, ki jo je potrebno izvesti, da stranki ponudimo celovito storitev itd. Ko govorimo o sestavu kot o seznamu vseh polproizvodov, imamo v mislih podroben načrt proizvoda ali krajše PBS ([39] Wright in Race, 2004; 226). Ko pa imamo v mislih sestav, zgrajen iz posameznih enot dela, pa govorimo o podrobnem načrtu dela za izvedbo projekta oz. krajše WBS. Po Wrightu in Racu ([39] 2004; 227) so prednosti uporabe PBS naslednje:

- osredotočenost na izložke dela,
- pomoč pri opredelitvi obsega proizvoda,
- je dobra osnova za načrtovanje,
- ponuja možnost dobre kontrole nad obsegom dela itd.

Glavne prednosti WBS pa so:

- je hierarhična sestav, za katerega navaja vse potrebne aktivnosti ali dejavnosti na projektu,
- člani projekt v manjše, logične enote dela, s čimer nam omogoča lažjo kontrolo,
- omogoča prepoznavanje ključnih trenutkov oz. mejnikov (*angl. milestones*) projekta itd.

Ta dva sestava sta med seboj tesno povezana. Iz PBS lahko določimo vse enote dela, ki so potrebne za izdelavo posameznega polproizvoda in obratno, iz WBS pa lahko določimo vse

učinke oz. njihove dele, ki so izložek določene enote dela. Na tem mestu je potrebno poudariti, da je izložek dela lahko tudi del storitve, zato lahko s PBS prikažemo tudi zgradbo storitev. V okviru te naloge se bomo osredotočili predvsem na urejen sestav vseh enot dela, se pravi na podroben načrt dela za izvedbo projekta oz. za izdelavo določenega poslovnega učinka (v nadaljevanju WBS).

Po Devauxu ([7] 1999; 73) mora imeti vsak projekt celovit in dobro razdelan načrt potrebnega dela (v nadaljevanju WBS). Odsotnost takšnega načrta lahko pripelje do prekoračitve predvidenih stroškov in rokov za izvedbo projekta, zelo verjetno pa vpliva tudi na kakovost poslovnega učinka projekta. Odsotnost pravtako vpliva na zaostajanje dejansko prislužene vrednosti (*angl. earned value*) za načrtovano vrednostjo ([47] Wikipedia: Earned value management, 2010). Če projekt nima WBS, hkrati pa je potrebno posredovanje zunanjega svetovalca, potem slednji ne more imeti vpogleda v to, koliko dela je potrebno narediti, niti v to, koliko dela je že bilo narejenega na projektu. Zelo verjetno je, da bo ta zunanji sodelavec najprej zahteval izdelavo dobrega WBS. WBS je pravzaprav ogrodje, na katerem temelji celoten projekt. Zelo dobrodošel je tudi na ostalih delih projekta ([10] Heldman in drugi, 2005; 127), predvsem pri načrtovanju časovne izvedbe, načrtovanju stroškov izvedbe projekta in predhodni pripravi ocen potrebnega časa ([39] Wright in Race, 2004; 228–229), stroškov in ostalih prvin.

Po Heldmanu in drugih ([10] 2005; 126) WBS služi tudi za komuniciranje glede obsega med vsemi udeleženci projekta: tako naročniki kot izvajalci. Najpomembnejši del projekta je njegova vsebina oz. obseg. In prav WBS je način, s katerim to vsebino spravimo v red (jo uredimo) in jo prikažemo v urejenem, hierarhičnem sestavu. WBS povezuje tudi 3 najpomembnejše strani projekta: vsebino, izdatke in čas. Žal je WBS pri vsakodnevnem ravnateljevanju projektov prej izjema kot pravilo.

Veliki projekti imajo ponavadi velik obseg, zato je zanje značilen zelo velik WBS, sestavljen iz 100 ali več gradnikov in umeščen v 3 ali več ravni. Izgradnja takšnega WBS je zahtevna in potrebuje veliko časa in znanja. S slednjim ne more razpolagati le en človek, zato je k izdelavi WBS smiselno povabiti različne strokovnjake ([13] Hermon, 1998; 25). Ravnatelj projekta ponavadi na podlagi obsega (oz. vsebine) projekta naredi prvo raven WBS, nato pa ostali člani projektne skupine ali celo zunanji strokovnjaki podrobneje razdelajo aktivnosti ali dejavnosti na posameznih ravneh. Slednji ponavadi postanejo tudi odgovorne osebe za načrtovanje izvedbe in samo izvajanje na posameznih ravneh. Tako WBS ne postane zgolj sestav potrebnega dela na projektu, temveč tudi sestav odgovornosti ([7] Devaux, 1999; 74).

Najpomembnejša prvina za pripravo WBS je prav gotovo seznam vseh zahtev naročnika (*angl. project scope statement*). Te zahteve natančno določajo celoten obseg projekta ([10] Heldman in drugi, 2005; 127) in služijo kot osnovni dogovor glede izložka projekta med naročnikom in izvajalcem projekta. Posledično je pomembna prvina za pripravo WBS tudi pravočasno dogovorjen in sprejet seznam zahtev za spremembo obsega projekta ([7] Devaux, 1999; 92). Govori se: »Če bi bil človek vedež, potem ne bi bil revež.« Slednji pregovor opisuje dejstvo, da v nekem trenutku ne moremo vedeti vsega, kar se bo v prihodnosti zgodilo. Zato je pomembno, da sta se tako naročnik kot izvajalec sposobna prilagajati obseg projekta glede na spremembe pri naročniku ali izvajalcu ali v okolju.

2.2.2 Ustvarjanje WBS s pomočjo razgradnje (*angl. decomposition*)

Izgradnje WBS se lahko lotimo na dva načina. Prvi način je razgradnja (*angl. decomposition*) zahtev. S tem načinom členimo osnovne zahteve projekta na manjše zahteve, bolj obvladljive skupke enot dela. Zamisel pri tem načinu je ta, da zahteve členimo na manjše enote dela tako dolgo, dokler teh enot dela ne moremo enostavno načrtovati, izvesti, spremljati, nadzorovati in tako zagotoviti, da je učinek enot dela skladen z zahtevami ([10] Heldman in drugi, 2005; 128). Najmanjše enote dela v WBS nam nato dajejo osnovo za:

- ocenjevanje časa, stroškov, prvin itd., potrebnih za opravljanje nalog,
- spremljanje ter nadzor nad opravljanjem nalog in
- dodeljevanje potrebnih prvin in odgovornosti na posamezno nalogo.

Drugi način je uporaba WBS iz predhodno že izvedenega projekta. Seveda je predpogoj za ta način podobnost med projekti in uspešna izvedba predhodnega projekta. Podrobnosti o obeh načinih sledijo v nadaljevanju.

Po Heldmanu in drugih ([10] 2005; 128) je postopek razgradnje sestavljen iz naslednjih 5 korakov:

1. Opredelitev vseh glavnih izložkov in aktivnosti (ali dejavnosti), povezanih z njimi. Glavne izložke opredelijo na podlagi mnenja enega ali več strokovnjakov, ki pregledajo osnovne zahteve projekta.
2. Določitev prve ravni sestava WBS in urejanje dela v ta sestav.
3. Razgradnja ravni WBS na podrobnejše ravni oziroma na podrobnejše skupke dela. Tako kot zahteve in izložki morajo biti tudi podrobnejši skupki dela opredeljeni na takšen način, da se jih da enostavno meriti in izložke dela enostavno primerjati z naročnikovimi zahtevami. Vsak skupek del mora jasno opisovati njegov izložek (proizvod, storitev ali kaj drugega) in mora biti kasneje dodeljen kot naloga ali aktivnost nekomu (posamezniku ali organizacijski enoti), ki bo odgovoren za uspešno izvedbo skupka del.
4. Dodelitev enoličnih številk ali zaznamkov posameznemu skupku del na vseh ravneh WBS.
5. Zadnji korak vsebuje preverjanje. Z njim želimo zagotoviti, ali so vsi sestavni deli WBS opredeljeni jasno in celovito. Zagotoviti je potrebno, da so v WBS prisotni vsi potrebni sestavni deli (ali skupki nalog) za izvedbo projekta in da je vsak sestavni del WBS res nujno potreben za zagotovitve izpolnitev zahtev naročnika.

Devaux ([7] 1999, 88) nam za razgradnjo zahtev v WBS podaja naslednjih šest priporočil:

1. Za poimenovanje posameznih aktivnosti (oz. sestavnih delov WBS) je potrebno uporabiti glagol s predmetom (npr. »preizkusi prilagodljivost«, »nariši sestav«). Na ta način že s poimenovanjem določimo, da sestavni del WBS predstavlja delo oz. aktivnost in za kakšno delo sploh gre.
2. Vsaka aktivnost (oz. sestavni del) mora imeti izložek (storitev ali (pol)proizvod ali kaj drugega). Na ta način lahko zelo preprosto ugotovimo, ali je delo opravljeno ali ne. Če že obstaja izložek, potem je delo narejeno. Če pa izložek še ne obstaja ali pa ni popolnoma ustvarjen, potem lahko ravnatelj projekta sodi, da delo še ni opravljeno.
3. Izložki vseh aktivnosti na neki ravni morajo biti skladni z izložkom aktivnosti, ki predstavlja njihovega »starša« na višji ravni. Povedano drugače: ko neko aktivnost razgradimo na več aktivnosti, morajo biti vsa dela, opravljena v teh aktivnostih, skladna s pričakovanim delom v glavni (razčlenjeni) aktivnosti.

4. Nobena aktivnost se ne sme oziroma ne more razgraditi na le eno aktivnost, temveč morata biti vsaj dve. Na ta način ustvarjamo nepotrebno podvajanje aktivnosti in s tem nepotrebno povečevanje sestava WBS.
5. Vsaka aktivnost mora biti dodeljena v izvedbo izključno enemu posamezniku ali eni organizacijski enoti ali enemu zunanjemu izvajalcu. Tako je zagotovljena jasna odgovornost za opravljeno delo in izložek. Če obstaja aktivnost, za katero sta odgovorni dve organizacijski enoti, potem je potrebno to aktivnost razgraditi na dve manjši aktivnosti in vsako dodeliti svoji organizacijski enoti. To je tudi eno od sodil, kako podrobno naj razgrajujemo posamezne ravni WBS. O tem nekoliko kasneje.
6. Zadnje priporočilo je: bolj kot je aktivnost tvegana oz. bolj kot je izvedba izložka negotova, na bolj podrobne ravni razgradimo to aktivnost. Tudi to je eno od sodil za določitev meje razgradnje. Bolj podrobno kot je razgrajena aktivnost, večja je verjetnost, da hitreje ugotovimo nastop tveganja in nato ustrezno ukrepamo.

Pri razgradnji (*angl. decomposition*) zahtev na WBS se vedno pojavlja vprašanje, kako globoko naj gremo z razgradnjo oz. pri kateri ravni se naj ustavimo. Sodila za določitev ustrezne ravni so:

- vsak zaokrožen skupek del (oz. sestavni del WBS, *angl. work package*), ki se v sestavi WBS več ne členi, mora imeti natanko eno odgovorno organizacijsko enoto, posameznika ali zunanjega izvajalca;
- raven tveganja za opravljanje posameznega skupka nalog;
- vsak končni skupek nalog naj ne bi obsegal več kot 80 ur dela ([7] Devaux, 1999; 90);
- vsak končni skupek nalog naj ne bi obsegal več kot 150 odstotkov časa, ki preteče med dvema poročanjema o napredku projekta ([7] Devaux, 1999; 90). Npr., če poročila o napredku projekta pripravljajo vsaka 2 tedna, potem skupek nalog ne sme trajati dlje kot 3 tedne;
- značilnosti projekta, kot so velikost, izložek itd.

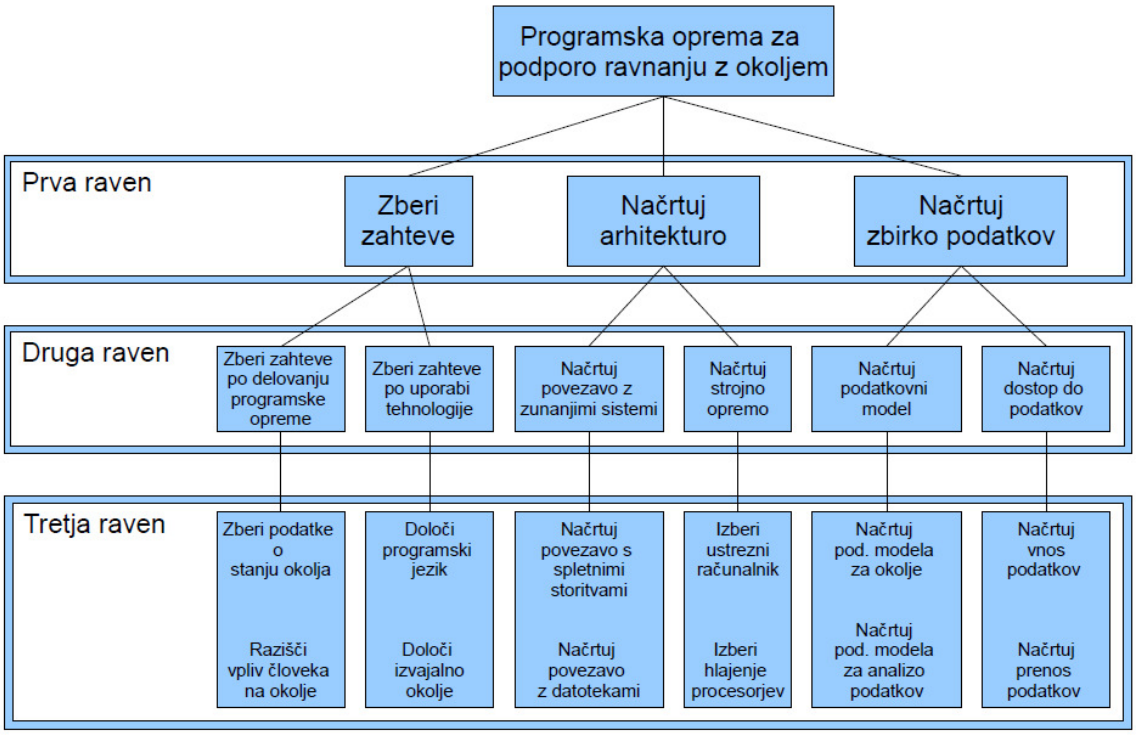
Kot vidimo, ima sestav WBS več ravni. Za Devauxa ([7] 1999; 76) imata največji pomen zgolj 2 ravni: najvišja (prva raven) in najnižja (zadnja raven). Na najvišji ravni je projekt, njegovo ime z oznako stroškovnega mesta, s čimer je potrjena ekonomska umeščenost za njegovo izvedbo. Najnižja raven pa vsebuje osnovne zaokrožene skupke dela (*angl. work packages*), ki kasneje kot aktivnosti predstavljajo dejansko delo, potrebno za izvedbo projekta. Na višjih ravneh ne izvedejo nobene konkretne naloge pri ustvarjanju poslovnega učinka projekta. Na najnižji ravni pa se vrši načrtovanje časovne izvedbe in dodeljevanje prvin k posameznemu skupku del. Seveda se takoj pojavi vprašanje, čemu služijo ostale ravni. Po Devauxu služijo trem namenom:

1. So sredstvo za povezovanje najvišje z najnižjo ravnjo. Slednje je doseženo z razgradnjo.
2. So osnova za zbiranje vseh informacij o opravljanju nalog na najnižji ravni. Te informacije zbirajo za potrebe poročanja srednjemu in višjemu ravnateljstvu podjetja.
3. So osnova za opredelitev stroškovnih, prihodkovnih, dobičkovnih in naložbenih mest za najnižje ravni.

Če povzamemo nekaj ugotovitev iz zgornjih poglavij, lahko zapišemo, da zaokroženi skupni del (*angl. work package*) predstavljajo najnižjo raven ([39] Wright in Race, 2004; 226). Kot je že bilo povedano, jih lahko dodelijo posamezniku, organizacijski enoti ali zunanjemu izvajalcu, s čimer se na njih prenese odgovornost za uspešno izvedbo. So osnova za ocenjevanje potrebnega časa, stroškov in poslovnih prvin.

Pri razgradnji je potrebno omeniti še možna sodila, po katerih razgradimo sestavo WBS. Sodila so lahko naslednja ([10] Heldman in drugi, 2005; 129):

1. Glavni izložki: Prva raven WBS se ustvari glede na glavne izložke projekta. Vsak glavni izložek predstavlja svojo vejo v sestavi WBS. Če, na primer, nameravaš odpreti trgovino, so glavni izložki lahko: določitev prostora trgovine, izgradnja trgovine, opremljanje trgovine itd.
2. Podprojekti: Prva raven se lahko ustvari tudi na podlagi podprojektov. Za vsak podprojekt naredi ravnatelj tega projekta svojo sestavo WBS. Pri tej delitvi se ravnatelj glavnega projekta ponavadi ne zaveda podrobnosti WBS za podprojekt in njegovega načrta za izvedbo. Kot primer takšnega WBS lahko navedemo gradnjo avtoceste, kjer so podprojekti: odstranjevanje stare ceste, načrtovanje trase, gradnja mostov, gradnja predorov itd.
3. Posamezne stopnje projekta: Veliko projektov se izvaja po posameznih, v vnaprej določenih stopnjah (*angl. phases*). Naprimer, pri razvoju programske opreme (glejte sliko 1) lahko kot stopnje opredelimo: zajem zahtev, analiziranje, načrtovanje, izvedba, preizkušanje in dostava programske opreme naročniku. Vsaka od teh stopenj se lahko na naslednji ravni členi na posamezne podstopnje (npr. načrtovanje arhitekture programske opreme, načrtovanje zbirke podatkov, načrtovanje uporabniškega vmesnika itd.).
4. Skupek več sodil (*angl. combination approach*): S tem načinom lahko uporabimo več sodil skupaj. Tako lahko za prvo raven razdelimo sodila glede na posamezne stopnje projekta in glavne izložke skupaj. Ta način lahko uporabimo tudi na nižjih ravneh.



Slika 1: Primer sestava WBS

2.2.3 Ustvarjanje WBS na podlagi WBS predlog iz predhodnih projektov

Razgradnja WBS je dolgotrajen, zahteven proces, ki zahteva vključitev velikega števila ljudi ([7] Devaux, 1999; 81–82), ki so po možnosti tudi strokovnjaki na svojih področjih. Proces razgradnje zahteva veliko komuniciranja, usklajevanja in sestankovanja. V ta proces je velikokrat potrebno vključiti tudi naročnika, ki se domnevno najbolje spozna na vsebino (oz. sestavo) WBS. Devaux ([7] 1999, 82) omenja celo posebnega ravnatelja, ki je zadolžen za razgradnjo WBS, o poteku razgradnje pa obvešča ravnatelja projekta. Seveda se temu procesu ne moremo izogniti, če delamo na projektu, ki ga projektna skupina (ali celotno podjetje) izvaja prvič. Lahko pa skrajšamo čas izdelave WBS v primeru, če smo predhodno (uspešno) izvedli projekt, ki je zelo podoben trenutnemu. V tem primeru govorimo o drugem načinu za izdelavo WBS.

Heldman in drugi ([10] 2005; 133) omenjajo uporabo predlog WBS iz predhodnih projektov. V trenutnem projektu lahko uporabimo večino strukture WBS, saj sklepamo, da bodo izložki projekta podobni. S tem privarčujemo veliko časa, ki bi ga sicer ravnatelj projekta porabil za izdelavo WBS. Kot primer takšnega projekta in WBS navaja Devaux ([7] 1999; 92–93) primer iz tovarne, ki je izdelovala igrače za otroke. Začeli so s projektom izdelave nove otroške igračke – punčke za dekleta. Bila naj bi manekenka, ki bi nosila veliko modnih dodatkov (torbico, (ogledalo ni modni dodatek), nakit itd.). Projekt izdelave te igrače je imel izdelano sestavo WBS z več kot 1200 aktivnostmi, ki so pokrivale načrtovanje in razvoj, oglaševanje, prodajo in razpošiljanje igrač. Ta WBS je nudil osnovo na pripravo ocen stroškov projekta. Naslednje leto se je okolje spremenilo. Prišla je zalivska vojna in zanimivi so postali mišičasti vojaki. Podjetje je pregledalo WBS za predhodno igračko in ugotovilo, da lahko za nov WBS (za mišičastega vojaka) uporabijo kar 900 od 1200 aktivnosti! In ne samo to. Za vseh teh 900 aktivnosti so imeli dejanske podatke (in ne ocene) za potreben čas in stroške, saj se te aktivnosti dejansko niso spremenile. Na ta način so bile ocene potrebnega časa in stroškov za izvedbo celotnega projekta veliko bolj natančne. Slednje je popolnoma skladno z ugotovitvami iz začetka tega poglavja, kjer je zapisano, da je pomemben vir podatkov za pripravo ocen zbirka podatkov o podobnih, že izvedenih projektih.

2.2.4 Predstavitev dodatnih načinov opisa zgradbe poslovnega učinka

Predhodno sem že zapisal, da WBS ni edini možen način opisa zgradbe poslovnega učinka. Predvsem v storitveni dejavnosti sta zelo uporabna naslednja načina:

- priprava načrta storitve (*angl. service blueprinting*) in
- opisovanje toka procesa (*angl. process flowcharting*).

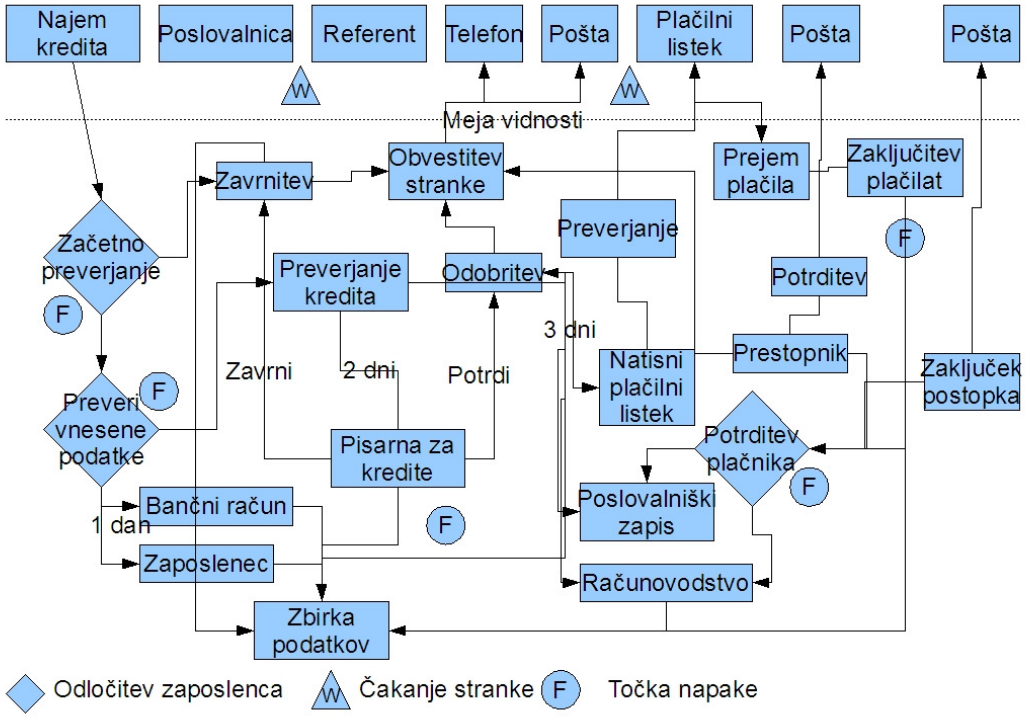
Pri razvoju nove storitve lahko hitro zaidemo v drage in dolge poskuse ([9] Fitzsimmons, 1998; 87), s katerimi želimo storitev dobro opredeliti in nato tudi začeti izvajati. S časom se je pojavila zamisel, da bi tudi na področje razvoja storitev uvedli inženirska znanja in pristope, kot je naprimer izvedba podrobnega, urejenega načrtovanja, katerega učinek je slikovit, urejenen načrt (npr. načrt arhitekture za večjo zgradbo). Na podlagi te zamisli je Lynn Shostack ([14] Hope in Mühlemann, 1997; 234) razvila načrt storitve (*angl. service blueprint*).

Načrt storitve je pravzaprav diagram toka izvajanja vseh aktivnosti, ki sestavljajo storitev. Aktivnosti imajo več različnih namenov ([9] Fitzsimmons, 1998; 87). Z nekaterimi obdelajo informacije, druge služijo za vzpostavitev stika s stranko in komuniciranje z njo, tretje

predstavljajo točke, kjer poteka sprejemanje odločitev med izvajanjem storitve itd. Načrt storitve ima v osnovi tri namene:

- služi za preučevanje izvajanja storitve, še preden se ta sploh začne opravljati. Po Fitzsimmonsu ([9] 1998; 87) lahko tako predvidimo izboljšave procesa vnaprej. Prav tako lahko s pomočjo načrta postopoma in urejeno členimo posamezne aktivnosti na podaktivnosti;
- po Hopu ([14] 1997; 234) omogoča prepoznavanje mest, pri katerih se lahko pojavijo težave (*angl. fail points*). Na podlagi ugotovljenih mest lahko pripravijo aktivnosti, ki odstranijo ali ublažijo te težave;
- razdeli aktivnosti storitve na dva dela ([14] Hopu, 1997; 234): na tiste, ki jih izvajajo ob stiku s stranko, in na tiste, ki jih izvajajo v ozadju, ko stranka ni več prisotna. Pri stiku s stranko je potrebno biti veliko bolj pazljiv, urejen, prijazen itd., saj je strankina predstava o storitvi v veliki meri odvisna prav od tega stika. Zato je zelo pomembno, da so aktivnosti, povezane s stranko, jasno vidne iz načrta.

Sestavni del načrta storitve oz. posameznih aktivnosti je povprečen (standarden) čas izvajanja posamezne aktivnosti. Ti časi nudijo osnovo za oceno časa celotne storitve. Lahko so podani s konkretno enoto ali s časovnim okvirjem (glejte sliko 2).



Slika 2: Primer načrta storitve ([9] Fitzsimmons, 1998; 88)

Po Hopu ([14] 1997; 234) je potrebno za izdelavo načrta storitev slediti naslednjim korakom:

1. Najprej je potrebno zaznati vse procese in postopke, ki so potrebne za izvedbo storitve, in jih vnesti v diagram (načrt). Raven podrobnosti predstavitve procesov in postopkov je odvisna od narave in zapletenosti storitve.
2. Nato je potrebno zaznati vsa mesta, pri katerih lahko nastanejo težave (*angl. fail points*). Za vsako takšno mesto je potrebno pripraviti proces ali postopek, ki bo odstranila to težavo, hkrati pa je potrebno izboljšati storitev tako, da se bo zmanjšala verjetnost ponovne pojavitve te težave.

3. Potrebno je določiti standardne čase za izvedbo posameznega postopka.
4. Ob samem izvajanju pa je potrebno spremljati, koliko strank se je poslužilo storitve in s kakšno kakovostjo.

Diagram toka procesa (*angl. process flowchart*) je naslednji možen način opisovanja zgradbe poslovnega učinka. Izvira iz proizvajalne dejavnosti, pri kateri so to orodje uporabljali za popis toka delovnih predmetov skozi delovna sredstva in mimo zaposlencev ter preoblikovanje delovnih predmetov ([33] Payne in drugi, 1996; 225). Pri opisovanju toka procesa se uporabljajo naslednje sestavine ([33] Payne in drugi, 1996; 225, [16] Johnstone in Graham, 2001; 160, [14] Hope in Mühlemann, 1997; 230):

- ○ = operacije (*angl. operation*): nakazujejo dele procesa, pri katerih se spremeni stanje delovnega predmeta;
- ⇨ = premikanje (*angl. transport*): določajo tiste dele procesa, pri katerih se dogaja premikanje prvin iz enega kraja v drugega;
- ▽ = skladiščenje (*angl. storage*): določa mesta skladiščenja prvin;
- D = zakasnitve (*angl. delay*): nakazujejo motnje v toku procesa;
- □ = kontrola (*angl. inspection*): so točke, na katerih se izvajajo pregledi nad procesom.

Kasneje je ta način popisovanja procesa v veliki meri začela uporabljati tudi storitvena dejavnost. Diagram toka proizvajalnega procesa ima veliko skupnega z načrtom storitve. Omogoča namreč možnosti za izboljšave, šibke točke procesa itd., vendar obstajajo tudi razlike. Glavna je ta, da se pri diagramu toka procesa bolj osredotočimo na razdaljo, ki jo mora zaposlenec ali stranka prehoditi, in čas, ki jo porabi za izvedbo posamezne aktivnosti, kot so zakasnitve, kontrola, operacije itd. ([9] Fitzsimmons, 1998; 136). Za potrebe storitvene dejavnosti so diagram nekoliko prilagodili (preglednica 1). Namesto elementa za skladiščenje uporabljamo korak stik s stranko (*angl. customer contact*).

	Razdalja	Čas	Vrste aktivnosti					Aktivnost
								Stranka zahteva račun
	10 m	0.5 min		⇨				Strežnik hodi
		0.5 min	○					Strežnik pripravi račun
	10 m	0.5 min		⇨				Strežnik hodi
		0.25 min				▽		Strežnik ponudi račun
	10 m	0.5 min		⇨				Strežnik hodi
		0.5 min					□	Stranka preveri račun in preveri kartico
	10 m	0.5 min		⇨				Strežnik se vrne
		0.25 min				▽		Strežnik vzame kartico
	10 m	0.5 min		⇨				Strežnik odide k blagajni
		0.5 min	○					Strežnik izpolni plačilni listek
		0.5 min	○					Strežnik obdela plačilni listek
		1 min					□	Strežnik preveri veljavnost kartice
	10 m	0.5 min		⇨				Strežnik hodi
		0.25 min				▽		Strežnik izroči plačilni listek stranki
	10 m	0.5 min		⇨				Strežnik hodi
		0.5 min	○					Stranka podpiše plačilni listek
	10 m	0.5 min		⇨				Strežnik hodi
		0.25 min	○					Strežnik vzame plačilni listek
	10 m	0.5 min		⇨				Stranka odide, strežnik hodi
Skupaj	**9 m**	**9 min**						

Preglednica 1: Primer diagrama toka procesa ([9] Fitzsimmons, 1998; 138)

2.3 Opredelitev osnovnih pojmov in metod

2.3.1 Opredelitev osnovnih pojmov

2.3.1.1 Predstavitev ugotovitev o preučevanju in razčlenjevanju dela v literaturi

Pri branju literature, ki se ukvarja s preučevanjem in razčlenjevanjem dela (npr. na projektu), bralec prej ali slej naleti na izzive. Slednji so posledica uporabe različnih poimenovanj pri opisovanju sicer enakih pojmov. To dejstvo vnaša v področje preučevanja dela veliko nejasnosti, neskladnosti in težav pri komuniciranju med samimi strokovnjaki. Omenjene nejasnosti so značilne tako za tujo kot za slovensko literaturo. Možno je, da je to lahko tudi eden izmed krivcev za slabše ravnateljevanje projektov v praksi. Vse skupaj kar kliče po urejenosti in skladnosti, po »inženirskemu« pristopu pri urejanju in poimenovanju področja za preučevanje dela.

Pa začnimo pri WBS oz. pri tehnični sestavi dela oz. projekta, kot besedo lepo poslovenita Rozman in Stare ([36] 2008; 142). WBS se členi na podprojekte, skupine aktivnosti na aktivnosti ([36] Rozman, Stare, 2008; 143). Aktivnost najprej opredelita kot ([36] 2008; 78) »... določen, smiselno vključen del projekta, za katerega izvedbo so običajno potrebni čas, ljudje in sredstva ...«, nato pa še kot ([36] 2008; 143) »... natančno določena delovna naloga z danim trajanjem, potrebnimi proizvodnimi prvinami za izvedbo in drugimi značilnostmi ...« Kot lahko razberemo, Rozman in Stare enačita pojem delovne naloge s pojmom aktivnosti. Žal v knjigi ni mogoče zaslediti natančne razmejitve med aktivnostjo in delovno nalogo. Kot bomo videli v nadaljevanju, je to (žal) prej pravilo kot izjema.

Če vzamemo v roke tujo (predvsem angleško) literuraturo, lahko ugotovimo, da se preučevanju dela napačno uporabljajo določeni izrazi. Tako lahko nek izraz nosi več pomenov ali pa lahko nek pojem najdemo opisan z večimi različnimi izrazi. Brez dodatnih pojasnitev! Naprimer, pri razčlenjevanju dela oz. projektov zasledimo naslednje izraze oz. poimenovanja:

- proces (*angl. process*),
- neprekinjena dejavnost ali operacija (*angl. operation*),
- storitev (*angl. service*),
- opravilo (*angl. task*),
- opravilo (*angl. job*),
- naloga (*angl. task*),
- aktivnost ali dejavnost (*angl. activity*),
- skupek aktivnosti (*angl. summary activity*),
- skupek dela (*angl. work package*),
- posamezna enota dela (*angl. work item*),
- skupek enot dela (*angl. work component*),
- korak dela (*angl. work step*),
- transakcija (*angl. transaction*),
- storitvena transakcija (*angl. service transaction*),
- vsebinsko zaokrožen skupek nalog (*angl. job*),
- tok dela (*angl. work flow*),
- neprekinjeno izvajanje dela (*angl. batch*)
- itd.

Ko zgoraj navedene termine preberemo tako »osamljene«, neumeščene v določeno vsebino, si jih lahko razlagamo na različne načine. V nadaljevanju bodo izrazi dodatno razloženi in umeščeni v ustrezen vsebinski okvir. Že na tem mestu pa je potrebno poudariti, da v literaturi le v redkih primerih srečamo ustrezno opredelitev terminov in razlik med njimi, še redkeje pa zasledimo enotno in skladno uporabo poimenovanj skozi celotno literaturo!

Slednje velja predvsem za opredelitve sestavnih delov večjih enot dela, kot so npr. projekt, proces, storitev ali funkcija (oz. neprekinjena dejavnost). Pa začnimo pri opredelitvi pojma opravilo (*angl. task* ali *angl. job*). Čeprav je pojem opravila »preprost« in vsem »razumljiv«, je njegovo opredelitev težko najti. Večina avtorjev opredeli opravilo kot osnovni sestavni del projekta, procesa ali storitve, po nekaterih je celo sestavni del aktivnosti. Hermone ([13] 1998; 26) piše, da se pri manjših projektih naredi ocena časa na ravni opravil, pri srednje velikih in velikih projektih pa se zaradi zahtevnosti naredi ocena na ravni aktivnosti. Johnstone in Graham ([16] 2001; 138) opredelita storitev oz. proces kot množico medsebojno povezanih opravil, prav tako Hope in Mühlemann ([14] 1997; 255). Slednja pri opisu metod za ocenjevanje časa celo opredelita, da je opravilo sestavljeno iz več aktivnosti, kar predstavlja protislovje njunim predhodnim trditvam. Pri uporabi termina opravila so še najbolj skladni in natančni Payne in drugi ([33] 1996; 233 in 236). Opravilo opredelijo na enak način Johnstone in Graham, Hope in Mühlemann, vendar, za razliko od njih, Payne in drugi skozi vse svoje delo uporabljajo termin opravilo, brez nepotrebnega omenjanja drugih »sorodnih« terminov, kot so (za ostale avtorje) npr. aktivnosti, delovna naloga itd. Kasneje bomo videli, da opravilo najbolje opredeli in umesti Mihelčič ([30] 2008; 366).

Naslednji pojem, ki v literaturi »buri duhove«, je pojem aktivnosti. Beremo lahko, da nekateri avtorji enačijo aktivnost z opravilom ([14] Hope in Mühlemann, 1997; 257), s čimer vnašajo v izrazoslovje določen nered. V istem delu lahko oba termina uporabljajo prevečkrat ali pa niso dosledni pri opredelitvi razlik med tema pojmoma. Johnstone in Graham opredelita aktivnost kot vmesni člen med procesom in opravilom ([16] 2001; 160). Enako naredita tudi Wright in Race ([39] 2004; 229). Fitzsimmons ([9] 1998; 88–89) ves čas piše o aktivnosti kot o osnovnem gradniku procesa ali storitve, pri čemer termina opravilo sploh ne uporablja. Korak naprej naredi Devaux, ki aktivnosti opredeli kot sestavne dele projekta (oz. WBS), vendar jih hkrati razdeli v dve skupini ([7] 1999; 78): skupek aktivnosti (*angl. summary activity*) in podrobne aktivnosti (*angl. detailed activity*). Podrobne aktivnosti omenja kot aktivnosti na zadnji ravni WBS, medtem ko skupke aktivnosti uporabi na vseh ostalih ravneh WBS. Podrobne aktivnosti celo enači s terminom posamezna enota dela (*angl. work item*). Pri delu Devauxa lahko zasledimo skladno rabo termina aktivnost skupaj s terminom posamezna enota dela. Zanimiv pogled na aktivnost imajo Heldman in drugi, ki pišejo ([10] 2005; 130), da za WBS aktivnost predstavlja premajhen sestavni del, zato priporočajo njeno uporabo zgolj pri majhnih projektih. Kot sestavni del WBS pri večjih projektih svetujejo uporabo večjih skupkov enot dela (*angl. work component*), kot je npr. podprojekt.

Navedimo še nekaj terminov, ki jih srečujemo. Johnstone in Graham pišeta o transakcijah. S slednjimi opredelita osnovni sestavni del za izvajanje storitve ([16] 2001; 164). Enak termin uporabita Hope in Mühlemann ([14] 1997; 259). Heldman in drugi veliko pišejo o že omenjenih skupkih enot dela (*angl. work component*). Slednje opredelijo kot osnovne gradnike WBS projekta (enako kot Fitzsimmons opredeli aktivnost!). Na najnižji ravni WBS nadomesti skupek enot dela nov termin: skupek dela (*angl. work package*) ([10] 2005; 130). Ta termin na enak način opredelita tudi Wright in Race ([39] 2004; 226). Heldman in drugi naredijo še korak naprej, ko skupek del opredelijo kot aktivnost ([10] 2005; 254). Kot lahko razberemo, je potrebno na področju poimenovanja uvesti red. Potrebno je poiskati in uporabiti

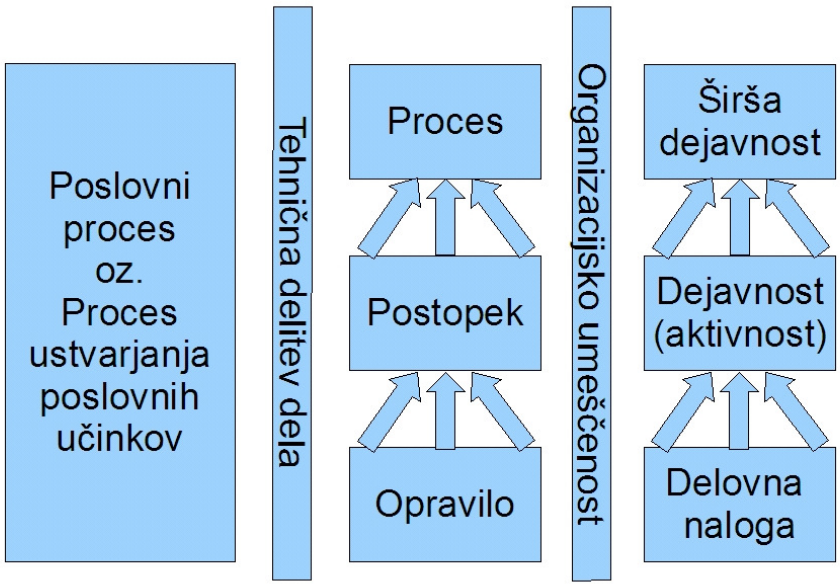
enotno in skladno poimenovanje, predvsem pa je potrebno odstraniti nepotrebna, »dvojna« poimenovanja istih pojmov pri delitvi dela. Ravno prava naloga za nadaljevanje.

2.3.1.2 Opredelitev pojmov pri razčlenjevanju dela

Najprej je potrebno opredeliti vlogo procesov (oz. kasneje projektov) v sami združbi. Širše gledano lahko v združbi govorimo o delovnem procesu ([29] Mihelčič, 1999; 271), ki predstavlja usmerjeno delovanje združbe k doseganju poslovnih ciljev. Delovni proces delimo na dva podprocesa:

- poslovni oz. izvedbeni proces in
- organizacijski proces.

Poenostavljeno si lahko prvi proces predstavljamo kot dejavnost družbe oz. proces ustvarjanja poslovnih učinkov iz nekoliko bolj tehničnega vidika, drugi proces pa kot proces, ki sproža sam poslovni proces s tem, ko za dejansko delovanje poslovnega procesa pripravlja predvsem ljudi in določa razmerja med njimi. Če poslovni proces nekoliko preučimo (tehnično razčlenimo), lahko ugotovimo, da je sestavljen iz posameznih podprocesov oz. postopkov, slednji pa so sestavljeni iz posameznih opravil (slika 3). Ob organizacijski umestitvi procesa v samo združbo (procesu določimo čas izvedbe, namen in prostor) pa začnemo govoriti o širši dejavnosti združbe (namesto procesa), o posameznih dejavnostih oz. aktivnostih (namesto postopkov) in o posameznih delovnih nalogah (namesto opravil). Na tem mestu je potrebno tudi zapisati ([30] Mihelčič, 2008; 65), da je v izvajanje posamezne dejavnosti vključenih več nosilcev (oz. zaposlencev).



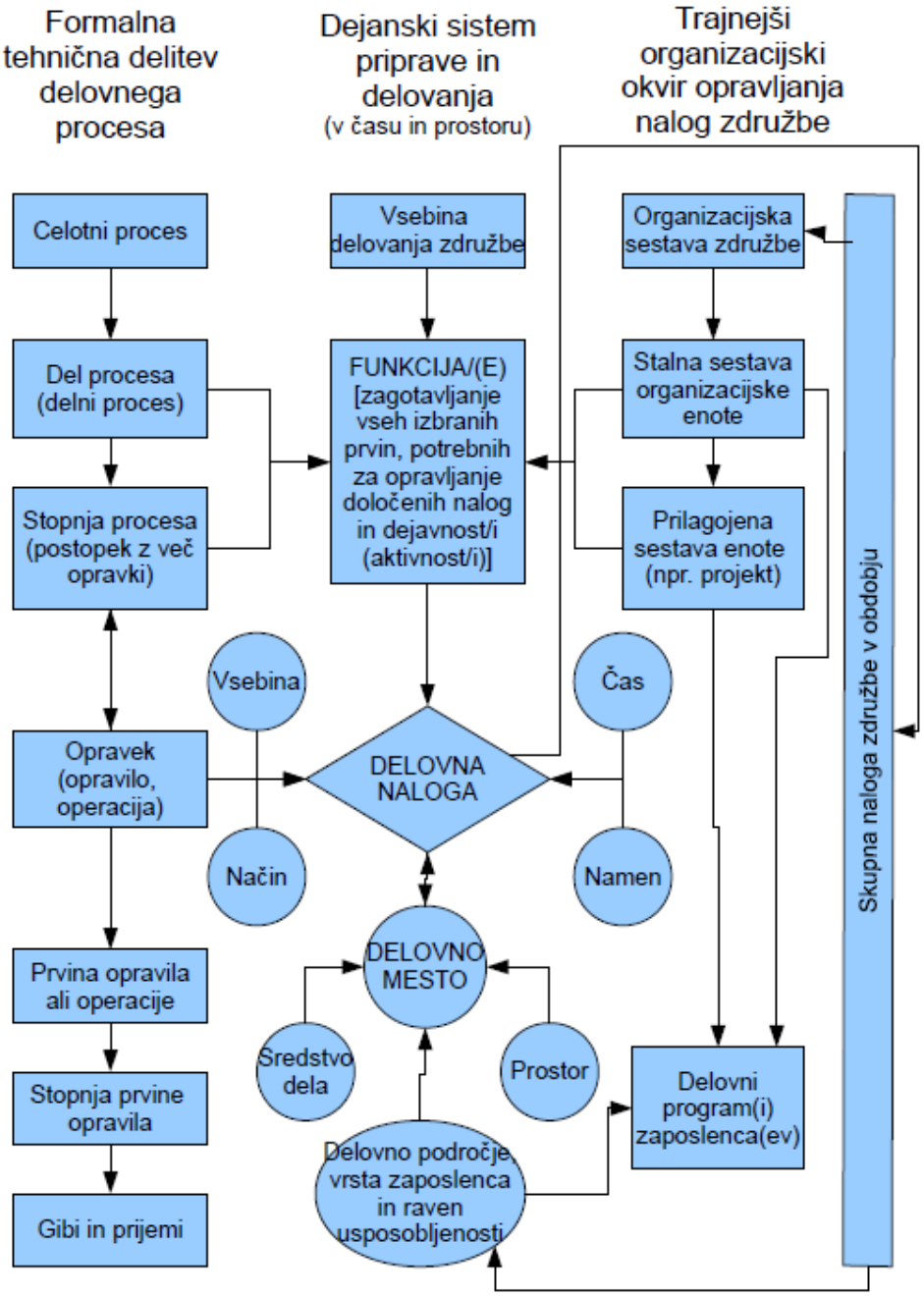
Slika 3: Poenostavljena predstavitev tehnične delitve dela in organizacijske umeščenosti poslovnega (oz. širše delovnega) procesa

Poslovni proces opiše Mihelčič tudi z naslednjimi besedami (povzeto, [29] 1999; 375): vsaka h gospodarskim ciljem usmerjena združba (npr. podjetje) opravlja v procesu družbene reprodukcije konkretno (družbeno) nalogo. To nalogo (proces) je mogoče deliti na delne, logično zaokrožene naloge (procese). Te lahko nadalje razstavimo na še podrobnejše in preprostejše skupine procesov ali stikov, dokler ne pridemo do stikov človeka z delovnimi predmeti in sredstvi pri ustvarjanju poslovnih učinkov (proizvodov, storitev). Te stike, ki sami po sebi nimajo ne časovne, ne prostorske in ne namenske razsežnosti, imenujemo opravila (ali opravki) ali dela. Običajno so opravki sestavljeni iz več prvin ali elementov, kot so stopnje (ali faze), gibi in prijemi. V praksi opravil pogosto ne razstavljamo naprej, tako da so to pretežno najbolj preprosti procesi, v katere razstavijo izbrani proces oziroma postopek dela. Opravilo (oz. opravek) je določen s štirimi določili:

- z zaposlencem neke vrste in ravni izobrazbe (npr. referent za stike s strankami, s 6. stopnjo izobrazbe iz smeri organizacijske vede),
- s sredstvom dela ustrezne vrste (npr. osebni računalnik),
- s kakovostjo ali vsebino dela, vključno s količino (npr. vnašanje novega komitenta),
- z načinom dela (npr. zajem podatkov o osebi preko uporabniškega vmesnika informacijskega sistema).

Vsako opravilo pa je lahko tudi delovna naloga, če opravek konkretiziramo, tako da mu določimo vlogo v procesu izpolnjevanja skupne naloge združbe (slika 4). Skupek vseh delovnih nalog združbe sestavlja delovni proces. Delovna naloga je najbolj podrobno opredeljena prvina delovnega procesa in je opredeljena s sedmimi določili ([30] Mihelčič, 2008; 368):

- z zaposlencem neke vrste in ravni izobrazbe (npr. referent Jaka Dolinar na bančnem okencu s 6. stopnjo izobrazbe smeri organizacijske vede),
- s sredstvom dela ustrezne vrste (npr. osebni računalnik z inventarno številko 123456 in z dvojedrnim procesorjem),
- s kakovostjo ali vsebino dela, vključno s količino (npr. vnašanje podatkov o novem komitentu),
- s prostorom, v katerem opravlja delo (npr. bančno okence št. 12),
- z načinom dela (npr. zajem podatkov o osebi preko uporabniškega vmesnika informacijskega sistema),
- z namenom, s katerim izvajamo opravke (npr. zagotoviti kakovostne podatke o poslovanju),
- s časom, v katerem mora biti opravljena (npr. 20 minut, v času od 10:25 do 10:45).



Slika 4: Poenostavljena shema sestavin delitve dela in opravljanja nalog(e) v združbi

Naj zaradi jasnega razumevanja na kratko povzamemo vse skupaj še enkrat. Poslovni proces si lahko najprej zamislimo (s tehničnega vidika) kot množico opravil, ki so med seboj povezana v postopke in nato v procese. V okviru tega pogleda se ne ukvarjamo s konkretnim časom, prostorom in namenom izvedbe procesa. Tukaj govorimo zgolj o ogrodju, o konkretni tehnični opredelitvi izvedbe posameznega procesa. Govorimo o vzorcu delovanja, ki je lahko za nek proces samo eden. Kot primer lahko navedemo opis in razčlenitev procesa odobritve kredita za fizične osebe na posamezna opravila. Ko pa proces organizacijsko umestimo, tako da procesu določimo čas, prostor in namen, pa govorimo o aktivnosti oz. posameznem primerku procesa. Teh je lahko za določen vzorec procesa seveda več. Kot primer lahko

navedemo konkretne izvedbe procesa odobritve kredita za vsako fizično osebo posebej. Če je za kredit zaprosilo 100 ljudi, se je zato izvedlo 100 primerkov procesa odobritve kredita. Za vsako fizično osebo svoj primerek procesa. Na tem mestu je potrebno poudariti, da hkrati z opravljanjem delovne naloge (dejavnosti ali širše dejavnosti) »v ozadju« izvedemo tudi opravilo (postopek ali proces).

Moč Mihelčičeve opredelitve poslovnega in organizacijskega (oz. širše delovnega) procesa dokazuje Vrhovec ([38] 2009). V svojem delu opisuje svojevrstno opredelitev poslovnega procesa izbrane združbe, jo opiše in hkrati tudi prikaže enostaven način preslikave med združbino in Mihelčičevo opredelitvijo delovnega procesa ([30] 2008; 96–97). Sklepamo lahko, da je Mihelčičeva opredelitev splošen način razumevanja delovanja poslovnega in (dela) organizacijskega procesa, ki se ga da uporabiti v poljubni združbi. To velja tako z vidika tehnične delitve dela kot z vidika organizacijske umeščenosti procesa. Kot zanimivost je potrebno omeniti še to, da Žvanut ([40] 2009; 26) imenuje tehnično razčlenjen proces (vzorec procesa) tudi referenčni proces.

Pri razvoju informacijskega sistema, ki bo podprl izvedbo nekega procesa, najprej obravnavamo tehnično delitev procesa. Poslovni analitiki in načrtovalci se ukvarjajo s procesi, postopki in opravili, ne pa s konkretno organizacijsko umeščenostjo procesa (npr. v banki). Šele ko je proces popolnoma podprt z informacijskim sistemom in se je banka odločila, da bo ta sistem uporabljala pri svojem poslovanju, in nič prej, lahko začnemo pisati o organizacijski umeščenosti procesa. Prav zaradi tega se bodo v nadaljevanju tega dela uporabljali predvsem termini (oz. izrazi), ki se nanašajo na tehnično delitev dela procesa (npr. postopek ali pa proces), in manj termini, ki se nanašajo na organizacijsko umeščenost procesa (npr. aktivnost ali delovna naloga).

2.3.2 Opis metod za ocenjevanje časa

2.3.2.1 Uvod

Ena izmed najtežjih nalog ravnatelja projekta je podajanje dobre ocene količine potrebnih prvin za uspešno izvedbo projekta. To trditev potrjujeta tudi Rozman in Stare ([36] 2008; 89), ki navajata, da je trajanje aktivnosti zaradi enkratnosti projekta in ponavadi tudi enkratnosti aktivnosti slednje skoraj nemogoče natančno napovedati. S tem problemom se soočimo tudi, če razpolagamo s podatki o izvajanju preteklih projektov. Omenjena avtorja navajata, da lahko ravnatelj projekta pridobi ustrezne približke o potrebnih prvinah iz podobnih preteklih primerov (projektov), iz raznih člankov in objav ter od oseb, ki so bile odgovorne za izvedbo posamezne aktivnosti.

Skozi čas so se na tem področju razvile različne metode za ocenjevanje potrebnih prvin oz. v našem primeru metode za ocenjevanje potrebnega časa za izvedbo aktivnosti in širše projekta. Kot je že zapisano v začetku tega poglavja, Hope in Mühlemann ([14] 1997; 255) delita metode v dve širši skupini: posredne (*angl. direct*) in neposredne (*angl. indirect*). Neposredne metode so tiste, s katerimi ocenjujejo potreben čas na podlagi predhodno opravljenih meritev na podobnih aktivnostih. Na ta način določijo povprečni čas izvajanja opravila. Šele ko opravilo organizacijsko opredelijo (npr. s projektom) in ga začno izvajati, slednje postane aktivnost s svojim, enkratnim in edinstvenim časom izvedbe. Posredne metode pa temeljijo na preučevanju zgradbe posameznega opravila. Vsako opravilo se razdeli na posamezne manjše enote, kot so: prvine opravila, stopnje in gibi ([30] Mihelčič, 2008; 366). Na podlagi

seštevkov posameznih časov sestavnih delov opravila se nato določi povprečni čas izvedbe opravila oz. kasneje aktivnosti. Tudi metoda ocenjevanja na podlagi mnenja strokovnjaka spada med posredne metode. V nadaljevanju bodo podrobneje predstavljene trenutno največkrat uporabljene metode za ocenjevanje potrebnega časa za izvedbo aktivnosti oz. projekta.

Na tem mestu je potrebno napisati še nekaj o napakah, ki nastanejo pri podajanju ocen. Ocene že po sami opredelitvi vsebujejo napake. Natančnost ocene je v veliki meri odvisna od količine in kakovosti informacij, s katerimi razpolaga oseba, ki pripravlja oceno. Več kot imamo kakovostnih informacij, bolj natančna je ocena časa ([10] Heldman in drugi, 2005; 257). Vir za te ocene so lahko izkušnje oseb na podobnih projektih, informacije, zbrane iz izvedb preteklih projektov, objavljene statistične informacije v literaturi ali pa zgolj (kar se zgodi v večini primerov iz vsakdana) informacije o konkretnem projektu, ki jih je ravnatelj s projektno skupino zbral pred začetkom izvajanja tega projekta. Namen te naloge je potrditi vzajemno, pozitivno povezavo med kakovostnimi informacijami, zbranimi iz izvedenih podobnih (preteklih) projektov, in natančnostjo ocene časa za izvedbo naslednjega projekta.

2.3.2.2 Ocena strokovnjaka (*angl. expert estimation*)

Verjetno najbolj uporabljena metoda za ocenjevanja potrebnih prvin (časa) je metoda ocenjevanja na podlagi mnenja strokovnjaka. Mnogi avtorji pišejo ([10] Heldman in drugi, 2005; [14] Hope in Mühlemann, 1997; [7] Devaux, 1999), da daje ta metoda najnatančnejše ocene (oz. izide). Strokovnjak je lahko kdorkoli iz projektne skupine. Pomembno je le, da ima dovolj strokovnega znanja o izvedbi opravila (oz. kasneje opravljanja aktivnosti), za katero podaja oceno. Dobri lastnosti te metode so:

- ocena je lahko podana razmeroma hitro, brez dolgotrajnih razčlenjevanj, opazovanj ali izračunov;
- s tem, ko oseba (strokovnjak) poda oceno za izvedbo določenega opravila, prevzame odgovornost za uspešno in pravočasno kasnejšo izvedbo aktivnosti ([7] Devaux, 1999; 104).

Vendar ima metoda tudi slabosti:

- glavna je ta, da oseba ponavadi poda svojo oceno brez stvarnih dokazov ([10] Heldman in drugi, 2005; 258). Ocene temeljijo na izkušnjah in občutkih. Slednje lahko vodi tudi do ocen z velikimi napaki, ki jih odkrijejo šele med samim uresničevanjem projekta, običajno takrat, ko je že prepozno;
- oseba, ki je podala oceno, lahko med opravljanjem aktivnosti ugotovi, da ocena ni dovolj natančna. Zgodi se lahko, da oseba npr. zaradi osebnih razlogov ne obvesti ravnatelja projekta o napaki v oceni, ampak želi izvesti aktivnost v predvidenem času. Zaradi tega se pojavita stres in napetost, ki vplivata na bistveno nižjo kakovost učinka aktivnosti ([14] Hope in Mühlemann, 1997; 260).

2.3.2.3 Ocena na podlagi podobnosti (*angl. analogous estimating*)

Osnova za oceno na podlagi te metode so podatki (oz. pridobljene informacije) o izvedbi podobnih projektov v preteklost. Pri tej metodi gre pravzaprav za različico metode ocenjevanja strokovnjakov ([10] Heldman in drugih, 2005; 258), vendar v tem primeru podajajo oceno na predhodno zbranih podatkih o podobnih projektih, s čimer pridobimo stvarne dokaze oz. razloge za dobro ocene. V najslabšem primeru služi ocena za osnovo, na

podlagi katere strokovnjak oblikuje boljšo oceno. Predpogoj za uspešno in učinkovito izvajanje te metode je trajno in sistematično zbiranje podatkov o uresničevanju projektov v združbi. Metoda je bolj učinkovita takrat, ko v okviru združbe skozi daljše časovno obdobje izvajajo večje število podobnih aktivnosti (oz. projektov). Kot primer navedimo delo gradbenega arhitekta. Slednji porabi za pripravo načrta srednje velike stanovanjske hiše standardnih oblik npr. 100 enot časa. Zelo verjetno je, da bo za pripravo novega načrta podobne stanovanjske hiše porabil približno 100 enot časa. Seveda, najverjetneje z manjšim časovnim odstopanjem. Dobre lastnosti te metode so še:

- dobra natančnost ocene ([7] Devaux, 1999; 106);
- dobro oceno lahko podajo že v začetnih stopnjah uresničevanja projekta, tudi v primeru, ko ravnatelj projekta nima na voljo vseh potrebnih informacij ([10] Heldman in drugi; 2005; 258);
- predhodno zbrani podatki služijo kot stvarni dokazi oz. dejstva, ki dokazujejo natančnost ocene.

Slabe lastnosti te metode so:

- podatke o opravljanju aktivnosti (oz. projektov) je potrebno sistematično zbirati, kar zahteva dodatno delo, s tem pa dodatno porabo prvin. Najtežje pri vsem tem je motivirati člane projektne skupine, da vestno beležijo podatke o opravljanju svojih nalog;
- natančno oceno je po tej metodi možno podati šele po tem, ko imamo na voljo podatke o vsaj enem izvedenem projektu.

Po Heldmanu in drugih ([10] 2005; 258) je najučinkovitejše ocenjevanje časa izvajanja projekta takrat, ko hkrati uporabljamo metodo ocenjevanja strokovnjaka z metodo ocenjevanja na podlagi podobnosti. Razlog za to je, da s slednjo metodo odpravimo večino slabih lastnosti metode ocenjevanja strokovnjakov.

2.3.2.4 Tritočkovno ocenjevanje (*angl. three-point estimates*)

Osnova te metode so tri predhodno podane ocene ([9] Fitzsimmons; 1998; 216). Govorimo o:

- optimistični oceni časa izvedbe; ta ocena predstavlja čas izvedbe opravila (kasneje aktivnosti), pri katerem ne predvidevamo težav ali zapletov;
- ocena pričakovanega časa izvedbe: to bi naj bila ocena povprečnega časa izvedbe opravila;
- pesimistični oceni časa izvedbe opravila; ta ocena vsebuje čas, porabljen za reševanje nenapovedanih zapletov.

Na podlagi teh treh ocen nato po določenem obrazcu ([36] Rozman in Stare, 2008; 108) izračunajo najverjetnejši čas izvedbe opravila (oz. kasneje aktivnosti). Poglavitna prednost te metode je predvsem preprostost in hitrost izračuna ocene najbolj verjetnega časa. Slabosti pa sta predvsem dokaj slaba natančnost in to, da morajo podati vse tri ocene ustrezni strokovnjaki. Na ta način imamo opravka tudi z vsemi slabimi lastnostmi metod ocenjavanja strokovnjakov.

2.3.2.5 Ocenjevanje časa na podlagi vnaprej določenih gibov

Ta metoda spada v skupino neposrednih metod za ocenjevanje časa. Uporabna je predvsem za ocenjevanje ponovljivih aktivnosti (oz. projektov), kot je npr. izdaja računa za stranko ([14] Hope in Mühlemann, 1997; 260). Ocenjevanje časa poteka tako, da izbrano aktivnost (oz.

opravilo, če še ni organizacijsko umeščeno) razčlenimo na posamezne manjše enote (npr. opravke, gibe itd.) in nato na podlagi opravljenih večkratnih meritev določimo povprečen čas za izvedbo manjših enot aktivnosti. Oceno časa celotne aktivnosti (oz. opravila) predstavlja vsota vseh povprečnih časov manjših enot.
Prednost te metode je njena učinkovitost in natančnost pri podajanju ocene za ponovljive aktivnosti, slabost pa je v njeni zoženi uporabnosti in dejstvu, da je potrebno za oceno časa narediti dolgotrajno preučevanje izvajanja aktivnosti. Po Paynu in drugih ([33] 1996; 234) lahko slednje traja tudi do 150 krat dlje, kot v povprečju traja izvajanje opazovane aktivnosti.

2.3.2.6 Proučevanje porabe časa (*angl. time study*)

Tudi študija časa spada med neposredne metode, osnovno enoto opazovanja pa predstavlja kar sama aktivnost ([33] Payne in drugi, 1996; 235). Oceno časa za izvedbo aktivnosti pripravijo na podlagi večkratnega opravljanja te aktivnosti (ne glede na osebo, ki opravlja aktivnost), nato pa se na podlagi pridobljenih meritev izračuna povprečni čas, ki predstavlja hkrati tudi povprečni čas izvedbe opravila ([14] Hope in Mühlemann, 1997; 255). Prednost te metode je učinkovitost pri oceni časa za ponovljive aktivnosti. Slabosti lahko najdemo v njeni omejeni uporabi, saj ni uporabna za oceno časa enkratnih aktivnosti. Prav tako je potrebno vložiti kar nekaj časa za zbiranje meritev.

Na tem mestu je potrebno navesti še nekaj drugih metod, ki jih lahko srečamo v literaturi. Te so: parametrsko ocenjavanje (*angl. parametric estimation* ([10] Heldman in drugi, 2005; 259)), metoda beleženja lastno porabljenega časa (*angl. self-logging* ([14] Hope in Mühlemann, 1997; 256)), metoda vzorčenja opravil (*angl. activity sampling* ([14] Hope in Mühlemann, 1997; 257)) itd.
Podrobnosti o teh metodah si lahko bralec prebere v navedeni literaturi.

3 Metode ocenjevanja časa pri razvoju programske opreme

3.1 Izhodišča metod ocenjevanja časa

3.1.1 Uvod

Začetki dejavnosti razvoja programske opreme segajo vse tja v 50. leta prejšnjega stoletja. Za začetno obdobje te dejavnosti je značilno to, da je bilo potrebno že razvito programsko opremo razviti znova takoj, ko se je na trgu pojavila nova vrsta računalnika z novostmi v strojni opremi. V tem času so se razvijalci in ravnatelji projektov zelo malo ukvarjali s samim ocenjevanjem in ravnateljevanjem vključno z načrtovanjem potrebnih poslovnih prvin (npr. potrebnega časa) pri razvoju programske opreme. S časom se je z razvojem prilagodljivejše strojne opreme razvijala tudi prilagodljivejša programska oprema, ob tem pa so v projekte razvoja programske opreme hkrati uvajali tudi ravnateljevanje vključno z načrtovanjem in bolj ali manj urejene pristope ter metode razvoja. K urejanju dejavnosti razvoja programske opreme je veliko pripomogla tudi kriza razvoja programske opreme (anlg. software crisis), ki se je pojavila med leti 1950 in 1980 ([48] Wikipedia, History of software engineering, 2009). Za to obdobje je značilno, da je veliko projektov prekoračilo porabo načrtovanih poslovnih prvih, kot so npr. čas ali finančna sredstva. V 90. letih so se začele pojavljati vedno bolj urejene in formalne metode razvoja programske opreme, orodja za podporo hitrejšemu razvoju, hkrati pa se je začelo vedno bolj razvijati tudi znanstveno in akademsko področje razvoja programske opreme. Med leti 1990 in 1995 so začeli množično uporabljati tudi novi informacijski medij - Internet, ki je na področje razvoja programske opreme uvedel novo razsežnost. Učinki omenjenih nabranih izkušenj in znanj iz področja razvoja programske opreme so tudi naslednje metode razvoja programske opreme ([49] Wikipedia, Software development process, 2010):

- slapovna (*angl. waterfall*) metoda,
- spiralna (*angl. spiral*) metoda,
- ponavljajoča in povečujoča (angl. iterative and incremental) metoda,
- agilne (*angl. agile*) metode, itd.

Kljub pridobljenim izkušnjam pa se pri razvoju programske opreme še vedno srečujemo s težavami v zvezi s prekoračitvijo porabe načrtovanih poslovnih prvin. Slednje si lahko razlagamo z dejstvom, da se dandanes povečuje tudi zapletenost programske opreme. Slednja mora biti vedno bolj prilagodljiva novim poslovnim zahtevam, podpirati mora vedno več poslovnih procesov, povezovati se mora z vedno večjim številom informacijskih sistemov in virov poslovnih podatkov, poslovnim uporabnikom mora ponujati dostop do podatkov preko različnih sporočilnih kanalov itd. Prav zaradi opisane povečane zapletenosti je potrebno dandanes vedno več časa porabiti za izvedbo stopenj analiziranja in načrtovanja programske opreme. Z dobro izvedeno stopnjo analiziranja dosežemo, da smo od naročnika uspeli pridobiti dovolj celovit nabor poslovnih zahtev, katerim mora zadostiti razvita programska oprema. Zaradi zapletenega in zahtevnega nabora poslovnih zahtev današnjih naročnikov lahko razmeroma majhna napaka oz. spregledana zahteva v stopnji analiziranja programske opreme povzroči velika odstopanja od načrtovanega časa izvedbe projekta in kakovosti programske opreme. Ključni učinek stopnje načrtovanja programske opreme predstavlja urejen, nedvoumen in strukturiran opis delovanja (oz. funkcij) programske opreme. Pri razumevanju zapletenosti delovanja programske opreme si lahko pomagamo tako, da z načrtovanjem to delovanje razdelimo na manjše, logično zaokrožene enote, katere lažje

obvladujemo. Brez učinkovitega načrtovanja je zelo težko zagotoviti nadzor nad razvojem zapletene programske opreme. Pri stopnjah analiziranja in načrtovanja programske opreme si lahko v ta namen pomagamo z uporabo ustreznih sodobnih orodij, kot so UML (*ang. Unified Modelling Language*) diagrami, listine primerov uporabe, entitetno relacijski diagrami itd.

Kot že napisano, postaja programska oprema oz. informacijska podpora za poslovne procesa širom sveta vedno bolj in bolj pomembna, zato postaja sposobnost za njen učinkovit, kakovosten, časovno kratek in ekonomsko sprejemljiv razvoj tekmovalna prednost. Ključen dejavnik za dosego te tekmovalne prednosti predstavlja kakovosten projektni načrt in učinkovito izvajanje slednjega. ([22] Kusumoto in ostali, 2004; 292, [25] McGarry in ostali, 2002; 86). Zelo pomembno je, da se pred uresničevanjem projekta poskuša napovedati in oceniti (ne)želene dogodke. Z izvajanjem ustreznih meritev jih lahko vnaprej zaznamo in vgradimo v projektni načrt. Ko govorimo o napovedovanju, imamo v mislih naslednje pojme: velikost, vložek truda (dela), čas razvoja, uporabljena tehnologija in kakovost. Najpomembnejša je ocena potrebnega dela, posledično pa ocena potrebnega časa za izvedbo tega dela ([22] Kusumoto in ostali, 2004; 292). Za namen priprave dobre ocene je bilo v preteklosti razvitih več načinov oz. metod, pri katerih je imel večinoma največjo težo obseg delovanja programske opreme. Več o tem nekoliko kasneje.

Ocene potrebnega dela oz. časa nosilci pripravijo že v zgodnjih stopnjah projekta, običajno takrat, ko projektna skupina še sploh ni vzpostavljena. Začetne ocene služijo za določitev časovnega in finančnega okvirja projekta. Zaradi pomanjkanja informacij na začetku projekta so te ocene pomanjkljive in ponavadi netočne. Zelo pomembno je, da se med samim izvajanjem projekta slednje sproti ustrezno popravljajo in prilagajajo spremembam projektnega plana ([25] McGarry in ostali, 2002; 86). Slabe ocene in napačno razumevanje samega procesa ocenjevanja običajno vodijo v neuspešno zaključene projekte. V veliko primerih so slabe ocene vzrok za načrtovane časovne roke, katerih sploh ni mogoče doseči. Vztrajanje pri takšnih rokih vodi v prekoračitve končnih rokov projekta, neustrezne zmogljivosti in slabo kakovost programske opreme. Razlogi za slabe ocene lahko najdemo v ([25] McGarry in ostali, 2002; 86):

- pomanjkanju izkušenj iz ocenjevanja,
- pomanjkanju zgodovinskih podatkov, na podlagi katerih se pripravijo začetne ocene,
- odsotnosti sistematičnega procesa ocenjevanja,
- nezmožnosti opredelitve ključnih opravil (z organizacijsko umeščenostjo pa delovnih nalog), ki predstavljajo osnovo za oceno,
- neutemeljena pričakovanja in predvidevanja ter
- nezmožnost prepoznavanja in upoštevanja neželenih dogodkov pri pripravi ocene.

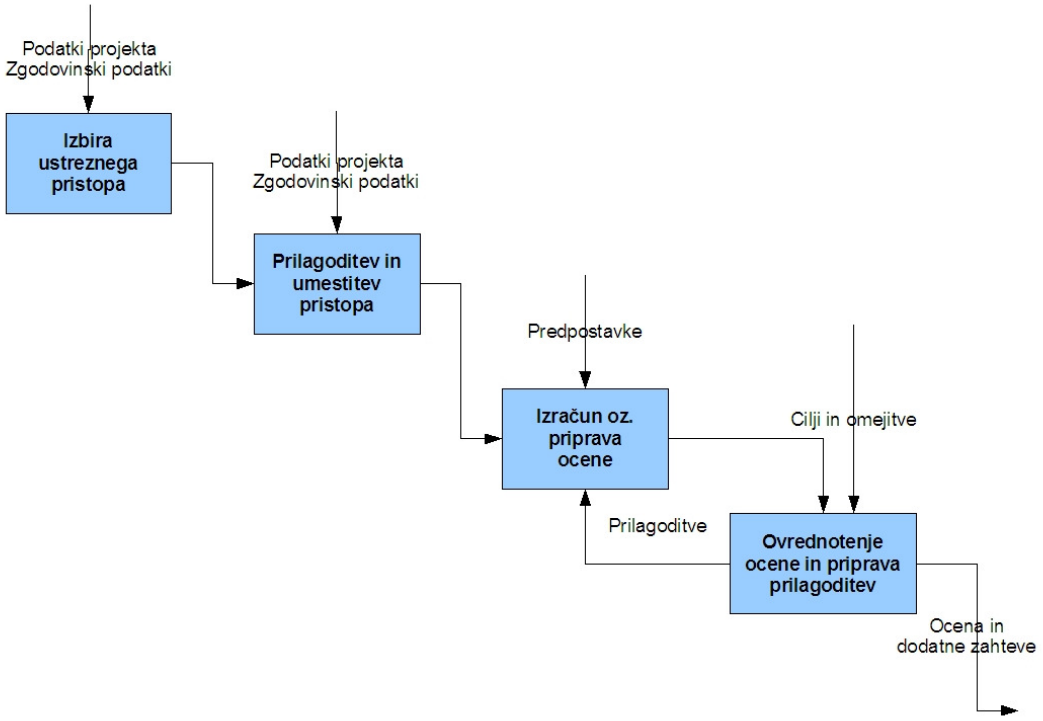
V preteklosti je bilo opravljenih že veliko raziskav na temo uspešnosti ocenjevanja potrebnega dela oz. časa za razvoj programske opreme. Raziskave kažejo, da so ocene ponavadi preveč optimistične in da so ocenjevalci preveč prepričani v pravilnost svojih ocen ([44] Wikipedia, Software development effort estimation, 2009). Kot primer prevelike gotovosti navajajo podatek, da je strokovnjak s področja razvoja programske opreme povprečno v 90 odstotkih oz. skoraj popolnoma prepričan, da je njegova ocena ustrezna. Zaključek projekta ponavadi pokaže, da bi morala biti njegova, skoraj popolna prepričanost nižja za 20 do 30 odstotkov. Kot je že zapisano v uvodu tega dela, poročilo združenja Standish group ([45] Standish group's CHAOS Report, 2009) kaže na to, da skoraj polovica projektov konča s prekoračitvijo časovnega ali finančnega obsega, z zmanjšanim obsegom ali kakovostjo, slaba četrtina projektov pa se zaključi neuspešno (so predčasno prekinjeni ali pa se njihov izložek ne uporablja).

Na tem mestu je potrebno poudariti, da je pri manjših projektih priprava ocene bistveno lažja in natančnejša oz. zahtevana natančnosti ni tako pomembna ([15] Issa in ostali, 2006; 2766). Pri velikih projektih pa napaka pri oceni postane veliko bolj izrazita. Zato je zelo pomembno, da se pri velikih projektih obvladuje zahtevana zapletenost in obseg programske opreme. Še toliko bolj, če združbe po vsem svetu v razvoj programske opreme letno vlagajo čez 10 milijard ameriških dolarjev ([11] Henry in ostali, 2007; 589).

Eden izmed pomembnih dejavnikov za zmanjševanje napake pri ocenjevanju potrebnega časa so podatki o času izvedbe preteklih projektov. O tem je bilo veliko že napisano v prejšnjem poglavju. Tu zapišimo še nekaj besed o tem, kako se tega opravila lotevamo pri razvoju programske opreme. Henry in ostali ([11] 2007; 600) pišejo, da si lahko podatke o predhodno izvedenih projektih predstavljamo kot spomin združbe (*angl. organization memory*), ki je lahko formaliziran, zapisan in viden vsem ali pa je skrit, nepopisan in shranjen le v glavah posameznikov. Pišejo, da obstaja pozitiven učinek med spominom združbe in uspešnim ravnateljevanjem ([11] 2007; 600). Pomembnost zgodovinskih podatkov poudarjajo tudi McGarry in drugi ([25] 2002; 85).

McGarry in drugi ([25] 2002; 89) so proces ocenjevanja pri razvoju programske opreme (in širše, tudi pri drugih procesih ustvarjanja poslovnih učinkov) razdelili na 4 stopnje (slika 5):

1) izbira ustreznega pristopa (oz. metode) ocenjevanja;
2) prilagoditev in umestitev pristopa izbranemu projektu;
3) izračun oz. priprava ocene;
4) ovrednotenje ocene in priprava ustreznih prilagoditev;



Slika 5: Proces ocenjevanja ([25] McGarry in ostali, 2002; 90)

Sestavni del procesa ocenjevanja so cenitvene prvine (*angl. estimators*) oz. meritve, ki so osnova za pripravo ocen ([25] McGarry, 2002; 87). Cenitvene prvine so lahko sestavljene tudi iz več meritev ali pa so izvedene iz izbrane meritve. Primeri cenitvenih prvin so: obseg delovanja učinka (*angl. functional size*), velikost učinka (*angl. product size*), vložek intenzivnosti dela (*angl. effort*), časovni razpored (*angl. schedule*), stroški (*angl. cost*), število napak (*angl. number of defects*), itd. V okviru te naloge se bomo posvetili predvsem cenitveni prvini »obseg delovanja učinka« in iz njega izpeljali oceno potrebnega časa.

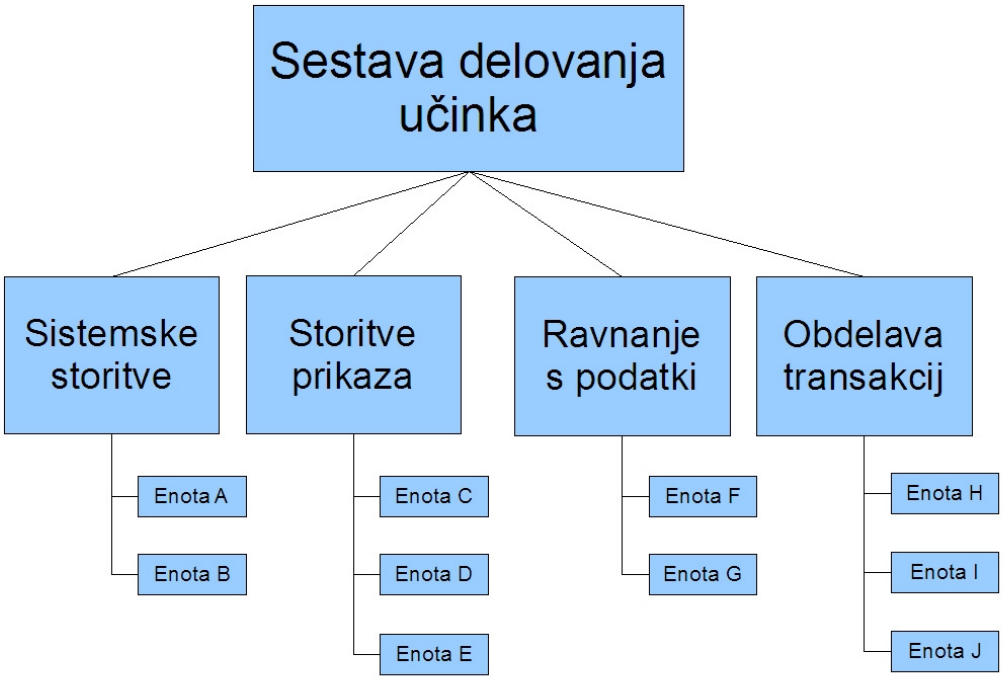
Razlika med cenitveno prvino »obseg delovanja učinka« in cenitveno prvino »velikost učinka« je ta, da s prvo ocenjujemo obseg delovanja učinka, kot ga zaznava zunanji opazovalec učinka. Ta zaznava zgolj njegovo delovanje, njegove dejanske zgradbe in velikosti pa ne pozna ([25] McGarry in ostali, 2002; 100). Obseg delovanja učinka je opredeljen v zahtevah naročnika programske opreme. Z drugo cenitveno prvino ucenjujemo dejansko velikost učinka, ki jo lahko določimo npr. s številom vrstic izvorne kode (*angl. lines of code*) programske opreme. Ocenjevanje časa na podlagi obsega delovanja programske opreme so se v preteklosti lotevali številni raziskovalci. Na tem mestu kot primere raziskovalcev navajam Herička in Živkoviča ([12] 2008; 772), Robiolo in Orosca ([35] 2008; 31), Kusumota in druge ([22] 2004; 292), Vinsena in druge ([37] 2004; 10) ter Cachia in druge ([6] 2005; 70). Cachia in drugi ([6] 2004; 10) navajajo, da obseg delovanja predstavlja osnovo za določitev ostalih cenilcev in da služi za osnovo pri pripravi ocene potrebnega časa, še preden je napisana prva vrstica programske opreme. Enakega mnenja so Kusumoto in drugi ([22] 2004; 292), ki navajajo, da je določitev obsega delovanja pomembna še zlasti za obsežno in zapleteno programsko opremo.

Vinsen in ostali ([37] 2004; 13) navajo WBS kot možen pristop k določitvi obsega delovanja programske opreme. Sestavni deli so tako lahko zaokrožene enote delovanja, ki jih delimo na manjše, smiselne enote. McGarry in drugi ([25] 2002; 52) opisujejo tri možne načine opredelitve WBS:

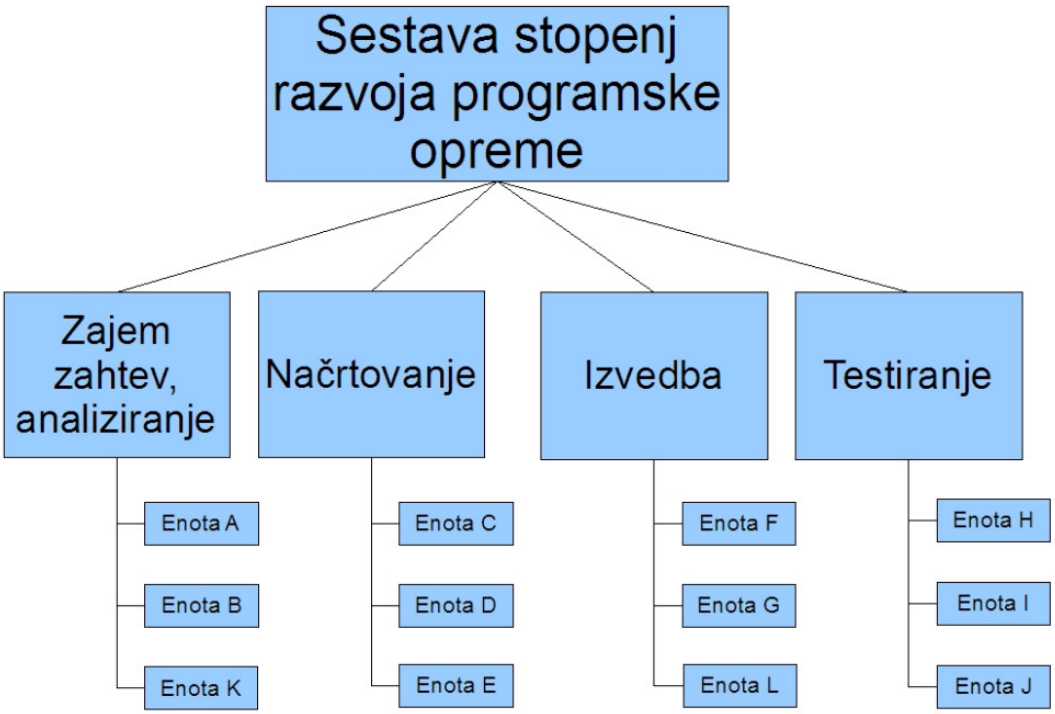
1. Zgradba sestavin (*angl. component structure*): Ta sestava deli programsko opremo na podsisteme, komponente in enote. Določa dejansko velikost programske opreme (slika 6).
2. Sestava delovanja učinka (*angl. functional structure*): Delovanja programske opreme razdeli na posamezne funkcije, podfunkcije in na koncu na najmanjše zaokrožene enote delovanja. Določa obseg delovanja programske opreme (slika 7).
3. Sestava stopenj razvoja programske opreme (*angl. activity structure*): Sestava prikaže sestavo procesa razvoja programske opreme po posameznih postopkih in na koncu opravilih (slika 8).

V nadaljevanju bomo podrobneje opisali pristope, s katerimi lahko opredelimo obseg delovanja programske opreme.

Slika 6: Zgradba programske in strojne opreme ([25] McGarry in ostali, 2002; 53)



Slika 7: Sestava delovanja programske opreme ([25] McGarry in ostali, 2002; 53)



Slika 8: Sestava stopenj razvoja programske opreme ([25] McGarry in ostali, 2002; 53)

3.1.2 Razvrstitev metod

Z ocenjevanjem velikosti, časa, napak itd. pri razvoju programske opreme se različni raziskovalci, univerzitetni učitelji in ljudje iz gospodarstva ukvarjajo že od 60. let prejšnjega stoletja ([44] Wikipedia, Software development effort estimation, 2009). V začetku se je večina raziskav osredotočila na iskanje metode, ki bi s pomočjo strogega, matematičnega modela ponudila ustrezno oceno. Danes obstaja veliko število takšnih metod, pri dejanskem razvoju programske opreme pa najbolj uporabljajo metodo COCOMO (*angl. constructive cost model*), katere začetki segajo v 70. leta prejšnjega stoletja. Sočasno so se začele pojavljati metode, ki temeljijo na drugačnih pristopih. Eden izmed pristopov je ocenjevanje časa na podlagi zahtev za programsko opremo. Danes sta tipični predstavnici tega pristopa metoda funkcijskih točk (*angl. function points*) in metoda točk primerov uporabe (*angl. use case points*). McGarry in ostali ([25] 2002; 90) navajajo, do lahko metode za ocenjevanje temeljijo na: zapletenih matematičnih modelih, preprostih aritmetičnih enačbah, množici pravil ali seznamu besedilnih navodil.

Zaradi naraščanja števila pristopov in metod za ocenjevanje je bilo potrebno na to področje uvesti ustrezen red. Slednje je bilo narejeno s pomočjo razvrstitev metod v ustrezne skupine. Zamisel ni nova; razvrstitve poznamo že v proizvajalni in storitveni dejavnosti, podrobneje pa so predstavljene v začetku tega dela. McGarry in ostali ([25] 2002; 90) predlagajo naslednjo razdelitev metod:

1. Parametrični modeli (*angl. parametric models*): metode, ki uporabljajo parametrične modele, predpostavljajo, da obstajajo razmerja med velikostjo, naporom, razporedom razvoja in kakovostjo programske opreme. Osnovo za določitev ocene predstavljajo teoretične predpostavke ali podatki o uresničevanju preteklih projektov. Tipičen primer te skupine je metoda COCOMO.

2. Modeli, ki temeljijo na aktivnostih (*angl. activity-based models*): osnovo teh metod predstavljajo ocene velikost, napor in/ali razpored izvajanja programske opreme in to za vsako zaokroženo enoto programske opreme oz. za vsako opravljeno nalogo projekta. Slednje so urejene v ustrezen sestav, kar omogoča pripravo ocene sklopov programa oz. aktivnosti projekta na višjih ravneh.
3. Analogije (*angl. analogy*): Na podlagi predhodnih projektov je mogoče narediti primerjave in pripravo ocen za nov projekt. To pristop predstavlja osnovo za metode te skupine.
4. Metode, ki temeljijo na preprostih razmerjih (*angl. simple estimation relationships*): Ta skupina metod temelji na poenostavitvi metod iz skupine parametričnih modelov. Za pripravo ocen se uporabljajo zgodovinski podatki iz uresničevanja predhodnih projektov ter preprosta enačba, ki opisuje glavne odvisnosti programske opreme.

Možnih delitev je veliko, v Wikipedii ([44]) je navedena delitev na 3 glavne skupine in na 7 podskupin. Jørgensen in Shepperd ([17] 2007; 34) navajata spet svojo. Več informacij o razvoju in raziskovanju metod za ocenjevanja velikosti, časa, truda, napak itd. si lahko bralec pridobi v delu omenjenih avtorjev.

V literaturi avtorji navajajo, da je največkrat uporabljena metoda za ocenjevanja časa metoda COCOMO. Slednja obstaja v več različicah, za namene te naloge pa se bomo osredotočili zgolj na osnovno. Slednja uporablja za pripravo ocene truda, potrebnega časa in ljudi naslednje enačbe:

$$Potreben\ trud = a_b(KLOC)^{b_b} \quad [\textbf{Enota}: število\ človek/mesec]$$

Enačba 1: Izračun potrebnega truda po metodi COCOMO

$$Čas\ razvoja = c_b(Potreben\ trud)^{d_b} \quad [\textbf{Enota}: število\ mesecev]$$

Enačba 2: Izračun časa razvoja po metodi COCOMO

$$Potrebno\ število\ ljudi = \frac{Potreben\ trud}{Čas\ razvoja} \quad [\textbf{Enota}: število]$$

Enačba 3: Izračun potrebnega števila ljudi po metodi COCOMO

Spremenljivka KLOC predstavlja oceno števila vrstic izvorne kode programske opreme v tisočih, konstante a_b, b_b, c_b in d_b pa so določene glede na vrsto projekta. Npr. vrednosti konstant v primeru projekta z majhno projektno skupino, ki je izkušena in ima dobro opredeljene zahteve s strani naročnika: a_b = 2.4, b_b = 1.05, c_b = 2.5 in d_b = 0.38 ([41] Wikipedia, COCOMO, 2009).

Kot primer izračuna ocene potrebnega časa lahko navedemo projekt izdelave programske opreme z 10.000 vrsticami izvorne kode. Glede na zgoraj podane enačbe dobimo oceno, da je za razvoj te programske opreme potrebnih 4,49 ljudi (glejte enačbe 4, 5 in 6).

$$Potreben\ trud = 2{,}4 * (10)^{1{,}05} = 26{,}93\ človek/mesec$$

Enačba 4: Primer izračuna potrebnega truda po metodi COCOMO

$$Čas\ razvoja = 2{,}5 * (26{,}93)^{0{,}38} = 5{,}99\ mesecev$$

Enačba 5: Primer izračuna časa razvoja po metodi COCOMO

$$Potrebno\ število\ ljudi = \frac{26{,}93\ človek/mesec}{5{,}99\ mesecev} = 4{,}49\ ljudi$$

Enačba 6: Primer izračun potrebnega števila ljudi po metodi COCOMO

Večina avtorjev navaja, da daje najboljše izide metoda ocene na podlagi mnenja strokovnjaka (*angl. expert estimation*). Podrobnosti si lahko priberete v delu Moløkkena in Jørgensena ([18] 2005). Nekateri avtorji poudarjajo, da je ta metoda tako uspešna zato, ker imajo strokovnjaki, ki pripravljajo oceno, v svojih glavah »shranjene« podatke o predhodno izvedenih projektih. Na podlagi teh podatkov in primerjav lahko nato pripravijo zelo dobre ocene. Učinkovitih metod je veliko. Predstavitev vseh metod bi preseglo namen te naloge, zato bodo v nadaljevanju podrobneje predstavljane le metode, ki so povezane z raziskavo v tem delu. Te metode so: metoda funkcijskih točk, metoda točk primerov uporabe in metoda poenostavljenih točk primerov uporabe. Razlog za izbiro omenjenih metod lahko najdemo v dveh lastnostih, ki so skupne vsem trem metodam:

- Oceno potrebnega časa za izvedbo projekta določijo na podlagi velikosti (oz. obsega) programske opreme ([22] Kusumoto, 2004: str. 292). O tem smo že pisali.
- Metode omogočajo pripravo dobre ocene potrebnega časa že v začetnih stopnjah projekta razvoja programske opreme.

Pomen zgodnje ocene opisujejo Vinsen in drugi ([37] 2004; 15), ki navajajo, da je zgodnja ocena pomembna predvsem za določitev potrebnih finančnih sredstev že na začetku projekta. Na podlagi teh podatkov lahko podajo sprejemljivo ceno za naročnika projekta, kar vodi do dobrih razmerij med izvajalcem in naročnikom. Robiola in Orosco navajata ([35] 2008; 30), da se ocena z običajnimi metodami pripravi prepozno; takrat, ko se je dejansko več ne potrebuje. Pišeta tudi, da je za hitro pripravo ocene potrebna zgodnejša in preprostejša metoda ocenjevanja ([35] 2008; 30). Kot primer takšne metode navajata metodo poenostavljenih točk primerov uporabe, ki jo bomo opisali kasneje. Zanimiv poskus iskanja metode za preprostejšo in zgodnejšo pripravo ocene predstavlja delo avtorjev Meli in Santillo ([26] 1999), ki opisujeta prilagojeno metodo funkcijskih točk za zgodnejše ocenjevanje.

3.2 Metoda funkcijskih točk (angl. function points analysis)

Začetki uporabe metode funkcijskih točk segajo v leto 1979, ko je njen avtor, g. Allan J. Albrecht, objavil prvi članek o metodi in njeni uporabi. Metodo je avtor s sodelavci v naslednjih 10 letih večkrat ustrezno prilagodil, hkrati pa je bilo ustanovljeno združenje uporabnikov te metode z imenom IFPUG (*angl. International Function Point User Group*). Glavni namen metode je bil odpraviti pomanjkljivosti takratnega ocenjevanja potrebnega časa

(oz. truda), ki je temeljilo na podlagi števila vrstic izvorne kode programa (*angl. lines of code – LOC*). Matson in drugi ([24] 1994; 275–276) navajajo, da metoda odpravlja naslednje dotedanje pomanjkljivosti:

- Težave z natančno opredelitvijo pojma »vrstica izvorne kode«. Matson in drugi navajajo ([24] 1994; 275), da obstaja kar 11 različnih metod, ki uporabljajo vrstico kode kot enoto za ocenjevanje, pri čemer jo vsaka opredeli po svoje.
- Odvisnost od uporabljenega programskega jezika. Težava se je (in se še) pojavljala zato, ker lahko npr. preprost matematični izračun v različnih programskih jezikih zapišemo z različnim številom vrstic kode. Metoda funkcijskih točk to (in prejšnjo) nepravilnost odpravlja, saj za merjenje velikosti uporablja pojem funkcijske točke, ki predstavlja osnovno enoto delovanja oz. funkcijo programske opreme. Slednja je neodvisna od izbrane tehnologije oz. programskega jezika.
- Ocenjevanje časa na podlagi vrstic kode je bilo možno narediti šele v času dejanskega programiranja ali pa predhodno glede na izkušnje strokovnjakov iz projektov (predvidevanje števila vrstic kode). Z metodo funkcijskih točk je mogoče narediti oceno že v začetnih stopnjah projekta in to brez predhodnih dolgoletnih izkušenj. Edina zahteva je ta, da funkcijske točke prešteje strokovnjak, ki se spozna na področje oz. poslovni sistem, katerega delovanje bo podprla programska oprema.
- Zadnja pomanjkljivost metode ocenjevanja števila vrstic izvorne kode je ta, da nehote poudarja zgolj stopnjo izvedbe (oz. programiranja) programske opreme. Slednja predstavlja le 10 do 15 odstotkov celotnega razvoja programske opreme. Metoda funkcijskih točk pa zaobsega tudi stopnje zajema zahtev, analiziranja, načrtovanja, testiranja (oz. preizkuševanja) in na koncu tudi stopnjo vzdrževanja programske opreme.

Metoda funkcijskih točk omogoča ocenjevanje velikosti in zapletenosti programske opreme na podlagi funkcijskih točk, ki predstavljajo zaokrožene enote delovanja (oz. funkcije) programske opreme ([24] Matson in drugi, 1994; 276). Zaokrožene enote delovanja niso odvisne od uporabljenega programskega jezika ali drugega orodja za razvoj programske opreme. Metoda je zasnovana za merjenje velikosti programske opreme, ki podpira predvsem delovanje poslovnih sistemov in ni primerna za znanstveno programsko opremo, ki je prvenstveno namenjena reševanju zapletenih matematičnih problemov. Funkcijske točke predstavljajo merilo, ki je neodvisno od programskega jezika, sistema za ravnanje z zbirkami podatkov, uporabljene strojne opreme ali katere koli druge tehnologije za obdelavo podatkov ([24] Matson in drugi, 1994; 276). Pomembna lastnost metode funkcijskih točk je ta, da omogoča spremljanje in nadzor izvajanja projekta razvoja programske opreme ([46]). Z njo je mogoče meriti produktivnost sodelujočih in napredovanje pri opravljanju nalog na projektu. Z njo je mogoče meriti tudi odstotek delovanja (oz. funkcij) sicer potrebne programske opreme, ki ga nismo zaznali v začetnih stopnjah razvoja programske opreme. S slednjim lahko merimo kakovost izvedbe zajema zahtev in analiziranja programske opreme. Večja kot je kakovost izvedbe zajema zahtev in analiziranja, manjši je odstotek nezaznanega delovanja programske opreme.

Osnovo metode predstavlja pet sestavin programske opreme ([46]):

- Zunanji vhodi (*angl. external inputs*): to so opravila, s katerimi okolje pošilja podatke v programsko opremo.
- Zunanji izhodi (*angl. external output*s): to so opravila, s katerimi programska oprema pošilja podatke v okolje.
- Zunanje poizvedbe (*angl. external inquiry*): to so opravila, ki izvajajo poizvedbe o podatkih znotraj programske opreme.

- Notranje logične datoteke (*angl. internal logical files*): to je množica podatkov, ki se nahaja izključno znotraj programske opreme.
- Zunanji datotečni vmesniki (*angl. external interface files*): to je množica podatkov, ki se nahaja izključno izven programske opreme (se pravi v okolju).

Pri pripravi ocene se vsaki od navedenih sestavin določi ena izmed treh vrednosti uteži: nizko, povprečno in visoko. Osnutek ocene števila funkcijskih točk se pripravi po enačbi 7 ([24] Matson in drugi, 1994; 276),

$$Število\ \ funkcijskih\ točk = \sum_{i=1}^{5}\sum_{j=1}^{3} w_{i,j}\cdot z_{i,j}$$

Enačba 7: Izračun števila funkcijskih točk

kjer spremeljivka $z_{i,j}$ predstavlja sestavino, spremenljivka $w_{i,j}$ pa utež za to sestavino. Da bi dobili končno oceno funkcijskih točk, je potrebno osnutek ocene ustrezno prilagoditi glede na dodatnih 14 dejavnikov programske opreme. Navedimo jih samo nekaj: zmogljivost, porazdelitev obdelave podatkov, učinkovitost dela končnih uporabnikov, zapletenost obdelav, ponovna uporabnost itd. Prilagoditev naredimo tako, da osnutek ocene pomnožimo s prilagoditvenim faktorjem c (glejte enačbo 8), ki omogoča povečanje ali zmanjšanje osnutka ocene za največ 35 odstotkov.

$$Prilagojeno\ število\ \ funkcijskih\ točk = c\cdot\sum_{i=1}^{5}\sum_{j=1}^{3} w_{i,j}\cdot z_{i,j}$$

Enačba 8: Izračun prilagojenega števila funkcijskih točk

Podrobnejši opis metode fukcijskih točk si lahko bralec prebere v navedeni literaturi ([46], [47]).

3.3 Metoda točk primerov uporabe (angl. use case points method)

Sodobna izpeljanka metode funkcijskih točk je metoda točk primerov uporabe. Slednjo največ uporabljajo v združbah, ki za razvoj programske opreme uporabljajo predmetno usmerjene (*angl. object oriented*) tehnologije ([4] Braz in Vergilio, 2006; 221). Uporabljajo jo tudi v podjetjih, pri katerih je sam razvoj programske opreme zasnovan na osnovni UML (*angl. unified modeling language*) diagramskih tehnik. Slednje uporabljajo za osnovo primere uporabe (*angl. use cases*), ki popisujejo delovanja sistema oz. popisujejo izvajanje posameznih poslovnih opravil, postopkov ali procesov. Osnovni gradniki primera uporabe so: uporabniki (oz. akterji), tokovi dogodkov oz. scenariji, predpogoji in popogoji, skice uporabniškega vmesnika itd. ([22] Kusumoto in drugi, 2004; 293). Pri ocenjevanju časa po tej metodi nas zanimajo predvsem uporabniki primerov uporabe in scenariji, iz katerih izluščimo transakcije. Slednje so opredeljene kot osnovna množica opravil (v literaturi se navaja termin aktivnost), ki jih izvedejo v celoti ali pa jih sploh ne izvedejo. Kot bomo videli kasneje, predstavlja število transakcij pomemben dejavnik pri pripravi ocene potrebnega časa.

Ker sta si metodi zelo podobni, je potrebno na tem mestu napisati tudi nekaj o razlikah. Pri obeh metodah predstavlja osnovo za pripravo ocene obseg delovanja programske opreme. Obe metodi prav tako omogočata pripravo ocene časa že v začetnih stopnjah projekta, poleg tega pa še nekateri avtorji ([1] Anda in drugi, 2005; 410) navajajo, da je mogoče (delno)

uporabiti metodo funkcijskih točk kar na primerih uporabe in obratno. Po Dreiemu in drugih ([8] 2005; 309) pa so glavne razlike naslednje:

- Metoda funkcijskih točk ne predpisuje nobenih formaliziranih listin, na podlagi katerih bi pripravili ocene. Metoda točk primerov uporabe pa to predpisuje, in sicer je osnovna listina »primer uporabe«, pri kateri je zahtevana določena oblika in zgradba. Slednje je zelo pomembno, če želimo razviti orodja, ki bi bila zmožna samodejno pregledati primere uporabe, iz njih razbrati ustrezne informacije in nato samodejno pripraviti oceno potrebnega časa.
- Metoda funkcijskih točk je že širše sprejeta in standardizirana. Slednje ne velja za metodo točk primerov uporabe, kar pa vpliva na to, da različna podjetja razumejo in uporabljajo metodo na različne načine. Posledica slednjega je ta, da so ocene med seboj težko primerljive.

Izračun ocene potrebnega časa po tej metodi izvedejo v šestih korakih ([1] Anda in drugi, 2005; 408, [22] Kusumoto in drugi, 2004; 293):

1. korak: izračun neprilagojene uteži uporabnikov (UAW – *angl. unadjusted actor weights*, glejte enačbo 9).

$$UAW = \sum_{i=1}^{n} Uporabnik_i \cdot Utež_i \ (n \ je \ število \ vseh \ uporabnikov)$$

Enačba 9: Izračun neprilagojene uteži uporabnikov

Vrsta uporabnika	Opis	Utež
Preprost	Programski vmesnik	1
Povprečen	Interaktiven vmesnik	2
Zahteven	Grafični uporabniški vmesnik	3

Preglednica 2: Uteži uporabnikov metode točk primerov uporabe

2. korak: izračun neprilagojene uteži primerov uporabe (UUCW – *angl. unadjusted use case weights*, glejte enačbo 10).

$$UUCW = \sum_{i=1}^{m} Primer \ uporabe_i \cdot Utež_i \ (m \ je \ število \ vseh \ primerov \ uporabe)$$

Enačba 10: Izračun neprilagojene uteži primerov uporabe

Vrsta primera uporabe	Opis	Utež
Preprost	3 ali manj transakcij	5
Povprečen	4 do 7 transakcij	10
Zahteven	Več kot 7 transakcij	15

Preglednica 3: Uteži primerov uporabe

3. korak: izračun neprilagojenih točk primerov uporabe (UUCP – angl. *unadjusted use case points*, glejte enačbo 11).

$$UUCP = UAW + UUCW$$

Enačba 11: Izračun neprilagojenih točk primerov uporabe

4. korak: izračun dejavnika tehnične zapletenosti in dejavnika okolja (TCF – *angl. technical complexity factor*, glejte enačbo 12, in *EF – angl. environmental factor*, glejte enačbo 13. Dejavnik »TFactor« (za tehnične dejavnike) izračunamo tako, da posameznemu dejavniku dodelimo eno izmed vrednosti med 1 in 5, nato slednja pomnožimo z utežjo dejavnika, na koncu pa vse zmnožke še seštejemo. Enako izračunamo dejavnik »EFactor« za okoljske dejavnike.

$$TCF = 0.6 + (0.01 * TFactor)$$

Enačba 12: Izračun dejavnika tehnične zapletenosti

Dejavnik	Opis	Utež
T1	Porazdeljen sistem	2
T2	Odzivnost ali zahtevana zmogljivost	1
T3	Učinkovitost končnega uporabnika	1
T4	Zapletena notranja obdelava podatkov	1
T5	Zahteva po ponovni uporabi izvorne kode	1
T6	Enostavnost namestitve	0.5
T7	Enostavnost uporabe	0.5
T8	Prenosljivost	2
T9	Enostavnost vnosa sprememb	1
T10	Sočasnost	1
T11	Vsebovanost posebnih varnostnih zahtev	1
T12	Omogočen neposreden dostop iz okolja	1
T13	Zahteva po posebnem izobraževanju uporabnikov	1

Preglednica 4: Tehnični dejavniki metode točk primerov uporabe

$$EF = 1.4 + (-0.03 * EFactor)$$

Enačba 13: Izračun dejavnika okolja

Dejavnik	Opis	Utež
E1	Poznavanje RUP metodologije	1.5
E2	Izkušnje iz razvoja programske opreme	0.5
E3	Izkušnje iz predmetno usmerjenih tehnologij	1
E4	Navzočnost vodilnega analitika	0.5
E5	Motivacija (oz .spodbude)	1
E6	Nespreminjajoče se zahteve	2
E7	Navzočnost sodelujočih s krajšim delovnim časom	-1
E8	Težak programski jezik	-1

Preglednica 5: Okoljski dejavniki metode točk primerov uporabe

5. korak: izračun prilagojenih točk primerov uporabe (UCP – *angl. use case points*, glejte enačbo 14).

$$UCP = UUCP * TCF * EF$$

Enačba 14: Izračun prilagojenih točk primerov uporabe

6. korak: izračun ocene potrebnega časa v enoti človek/ura (E – *angl. effort*, glejte enačbo 15). Pri izračunu uporabimo podatek o produktivnosti »PHPerUCP«. V literaturi navajajo ([23] Manogheghi in drugi, 2005; 308), da je njegova vrednost tipično med 20 in 36 človek/ur na posamezen primer uporabe.

$$E = UCP * PHPerUCP$$

Enačba 15: Izračun ocene potrebnega časa v enoti človek/ura

Podrobnejši opis metode točk primerov uporabe si lahko bralec prebere v navedeni literaturi ([1], [21]).

Kot primer izračuna ocene potrebnega časa lahko navedemo projekt izdelave programske opreme, katere delovanje (oz. funkcije) je opisano s preprostim in zahtevnim primerom uporabe, uporabljata pa jo povprečen ter zahteven uporabnik. Zaradi poenostavitve izračuna ima vsak dejavnik tehnične zapletenosti in dejavnik okolja vrednost 1, produktivnost pa znaša 20 človek/ur. Glede na zgoraj podane enačbe dobimo oceno, da je za razvoj te programske opreme potrebnih 461,8 človek/ur (glejte enačbe 16, 17, 18, 10, 20, 21 in 22).

$$UAW = 2_{(povprečen)} * 1 + 3_{(zahteven)} * 1 = 5$$

Enačba 16: Primer izračuna neprilagojene uteži uporabnikov

$$UUCW = 5_{(preprost)} * 1 + 15_{(zahteven)} * 1 = 20$$

Enačba 17: Primer izračuna neprilagojene uteži primerov uporabe

$$UUCP = 5 + 20 = 25$$

Enačba 18: Primer izračuna neprilagojenih točk primerov uporabe

$$TCF = 0.6 + (0.01 * 14) = 0{,}73$$

Enačba 19: Primer izračuna dejavnika tehnične zapletenosti

$$EF = 1.4 + (-0.03 * 4{,}5) = 1{,}265$$

Enačba 20: Primer izračuna dejavnika okolja

$$UCP = 25 * 0{,}73 * 1{,}265 = 23{,}09$$

Enačba 21: Primer izračuna prilagojenih točk primerov uporabe

$$E = 23{,}09 * 20\ človek/ur = 461{,}8\ človek/ur$$

Enačba 22: Primer izračuna ocene potrebnega časa v enoti človek/ura

3.4 Metoda poenostavljenih točk primerov uporabe

3.4.1 Uvod

Temelje te metode sta postavila Robiola in Orosco ([35] 2008). Pri iskanju nove metode sta si zadala nalogo, da poiščeta preprostejšo in natančnejšo metodo, kot je metoda funkcijskih točk. Z novo metodo sta želela doseči, da:

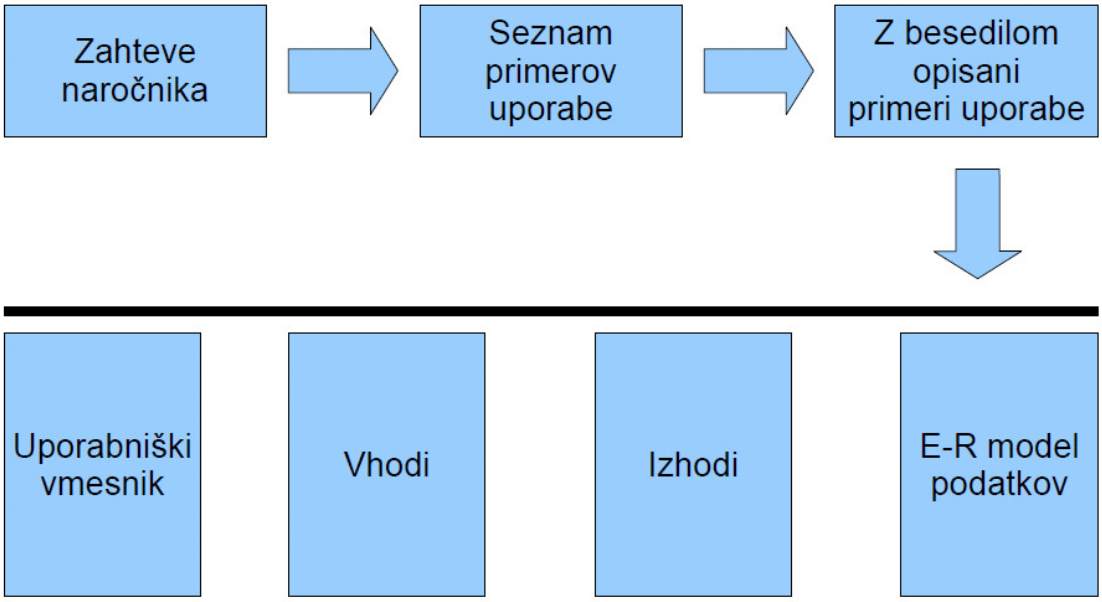
1) se ocenjevanje potrebnega časa lahko izvede dovolj zgodaj na projektu in
2) da se zmanjša velikost napake ocene potrebnega časa.

Kot glavne pomanjkljivosti metode funkcijskih točk avtorja navajata medsebojno odvisnost sestavin programske opreme, kot so zunanji vhodi in zunanji izhodi, ter zastarelost in neprilagodljivost metode glede na sodobne razvojne procese itd. Avtorja sta pod drobnogled vzela tudi metodo točk primerov uporabe. Kot glavne pomanjkljivosti te metode navajata:

- omejitev zaradi majhnih zalog vrednosti posameznih sestavin (npr. uporabnik je opredeljen kot preprost, povprečen in zahteven). Pri takšni zalogi vrednosti lahko zgolj medsebojno primerjamo posamezne sestavine, ne moremo pa natančno izmeriti velikosti njihovih razlik (npr. za koliko je nek uporabnik zahtevnejši od drugega);
- veliko stopnjo napake pri ocenjevanju potrebnega časa, ki kaže na to, da ta metoda še ni dovolj zrela za uporabo v praksi ([1] Anda in drugi, 2005; 415);

Novo metodo sta osnovala na primerih uporabe, vendar na nekoliko prilagojen način, saj sta velikost primerov uporabe izmerila s samo dvema neodvisnima sestavinama: transakcijo in entiteto. Robiola in Orosco menita, da kljub omenjenim pomanjkljivostim primeri uporabe vseeno predstavljajo dober način za popis obsega delovanja programske opreme. Kot glavne prednosti primerov uporabe navajata ([35] 2008; 31), da:

1) je njihova uporaba mogoča tudi pri predmetno usmerjenem razvoju;
2) jih uporabljajo širom po svetu;
3) primeri uporabe predstavljajo eno izmed osnovnih sestavin razvoja programske opreme – RUP (angl. *rational unified process*);
4) so v primerih uporabe osnovni opisi delovanja programske opreme na voljo že v zgodnjih stopnjah razvoja (slika 9). Za izračun funkcijskih točk bi v tem primeru morali počakati na listine (oz. učinke), ki opisujejo uporabniški vmesnik, vhode, izhode in model podatkov, s čimer bi s pripravo ocene čakali vse tja do zaključka stopnje načrtovanja programske opreme;



Slika 9: Zaporedje učinkov (oz. listin), ki jih običajno ustvarijo pri razvoju ([35] Robiola in Orosco, 2005; 33)

Kot rečeno, predstavljata sestavini »transakcija« in »entiteta« osnovni merili za določitev velikosti oz. zahtevnosti primera uporabe. Uporabo slednjih avtorja utemeljita z dejstvom, da lahko vsako programsko opremo opazujemo z dveh vidikov: z vidika delovanja (oz. funkcij) in z vidika podatkov. Kot osnovno zaokroženo enoto delovanja avtorja opredelita transakcijo, osnovno zaokroženo enoto trajno shranjenih podatkov pa opredelita kot entiteto ([35] Robiola in Orosco, 2008; 34–35).

Ko že pišemo o primerih uporabe in njihovi vlogi pri popisovanju delovanja programske opreme, je treba napisati še nekaj besed o razmerju med primeri uporabe in načrtom potrebnega dela za izdelavo programske opreme oz. WBS (*angl. work breakdown structure*). Primere uporabe (PU) si lahko predstavljamo tudi kot mero za oceno potrebnega dela. Bolj kot je delovanje programske opreme, opisano v PU, zapleteno, več dela bo potrebnega za razvoj te programske opreme. Vinsen in drugi ([37] 2004; 13) navajajo, da se na projektih razvoja programske opreme za določitev in popis obsega delovanja pogosto uporablja WBS, pri čemer posamezni, podrobno razdelani sklopi naročnikovih zahtev predstavljajo končne gradnike (oz. liste) v sestavi WBS. Kot primere podrobno razdelanih zahtev navajajo tudi primere uporabe, ki jih na višji ravni povežejo v splošnejše zahteve naročnika (slika 7). Če imamo na projektu na voljo WBS skupaj s primeri uporabe, lahko preštejemo transakcije in entitete tako, da za vsak končni list WBS določimo število transakcij in entitet, nato pa seštejemo število vseh transakcij in entitet po posameznih končnih listih. Praktičen primer lahko bralec preuči v nadaljevanju.

3.4.2 Opredelitev transakcije

Avtorja navajata ([35] 2008; 33) Jacobsonovo opredelitev primera uporabe, ki piše, da je primer uporabe poseben način uporabe informacijskega sistema oziroma izvajanje posameznega sklopa njegovega delovanja. Vsak primer uporabe je sestavljen iz celovitega niza dogodkov, ki jih sproži zunanji uporabnik, hkrati pa primer uporabe opisuje medsebojni

vpliv med zunanjim uporabnikom in informacijskim sistemom. S sistemskega vidika lahko rečemo, da je primer uporabe popoln tok delovanja informacijskega sistema, kar lahko opredelimo tudi kot transakcijo. V enem samem primeru uporabe je lahko opisana ena ali več transakcij. Robiola in Orosco transakcijo razdelita na dva dela ([35] 2008; 35):

- na začetno vzpodbudo oz. dražljaj, ki ga sproži zunanji uporabnik in
- odgovor sistema na ta dražljaj.

Odgovor sistema si lahko predstavljamo tudi kot izvedbo zaporedja sprememb v sistemu, ki uporabniku ustvarijo novo dodano vrednost. Namen tega dela namreč ni določiti najpravilnejše opredelitve transakcije. Pojem transakcije je zelo širok in se ga uporablja v različnih dejavnostih, vse od računalništva, bančništva, prodaje nepremičnin, zavarovalništva, zdravstva itd. V okviru raziskovalnega dela te naloge bo navedena opredelitev transakcije, za katero avtor tega dela meni, da mu najbolje koristi pri štetju transakcij iz listin primerov uporabe. Več o tem v nadaljevanju.

Robiola in Orosco v svojem delu navajata nabor pravil, s katerimi lahko na enostaven način preštejemo število transakcij. Njuna osnovna zamisel temelji na uporabi slovničnega stavčnega razčlenjevanja, ki predstavlja sistematičen, dobro znan, predvsem pa objektiven način prepoznavanja vsebin v besedilih. Tukaj je potrebno poudariti, da v delu žal ni jasno opredeljene meje med tem, kateri glagoli določajo dražljaj (oz. transakcijo) in kateri ne. Natančnejša opredelitev je prepuščena avtorju, več o tem pa si lahko bralec prebere v delu Robiole in Orosca ([35] 2008; 35). Na podlagi stavčnega razčlenjevanja sta opredelila naslednji temeljni pravili:

- vsak predmet (kot stavčni člen) v stavku lahko predstavlja uporabnika in
- vsak glagol lahko predstavlja začetni dražljaj oz. transakcijo.

Število tako preštetih dražljajev predstavlja tudi število vseh transakcij v primeru uporabe, število vseh transakcij programske opreme pa predstavlja vsota vseh transakcij po posameznih primerih uporabe. Določitev obsega (velikosti) delovanja programske opreme se lahko tako izračuna po spodnji enačbi:

$$Obseg\ delovanja = \sum Število\ transakcij\ v\ primeru\ uporabe$$

Enačba 23: Izračun obsega delovanja programske opreme s pomočjo števila transakcij

Pri izračunu obsega celotnega delovanja avtorja navajata dodatna pravila ([35] Robiola in Orosco, 2008; 35), ki zagotavljajo pravilno štetje transakcij:

1) Kadar listina »primer uporabe« (v nadaljevanju PU) razširja oz. uporablja drugi PU, se število transakcij prvega PU prešteje neodvisno od števila transakcij že obstoječega PU. Razloge za uvedbo tega pravila lahko najdemo v dejstvu, da se transakcije med PU-ji ne smejo ponavljati, ampak se mora drugi PU zgolj sklicevati na transakcijo v prvem PU. V tem primeru se sklicevanje na transakcijo ne šteje kot dodatna transakcija. Na ta način se izognemo dvakratnemu štetju transakcij.
2) Dražljaji zunanjih uporabnikov se obravnavajo (oz. štejejo) na enak način, ne glede na to, ali so del opisa osnovnega delovanja (oz. scenarija) ali dodatnega delovanja, oziroma ne glede na to, ali govorimo o osnovnem (primarnem) uporabniku sistema ali uporabniku, ki le občasno uporablja delovanje. Razlog za uporabo tega pravila lahko najdemo v dejstvu, da izvedba transakcije ni odvisna od vrste uporabnika, ki jo bo izvedel, niti od tega, ali predstavlja osnovno ali le občasno enoto delovanja. V vsakem primeru jo je potrebno izvesti in s tem tudi informacijsko podpreti s programsko opremo, kar pa seveda vpliva na velikost delovanja programske opreme.

3) Če v podrobnem opisu PU obstaja en ali več grafičnih diagramov (npr. diagram aktivnosti), ki opisujejo tok delovanja, je potrebno tudi pri njih določiti število dražljajev (oz. transakcij), ki jih prožijo zunanjih uporabniki. V tem primeru se lahko uporabi prilagojena oblika stavčnega razčlenjevanja, lahko pa se uporabijo tudi drugačni pristopi. Tudi pri tem pravilu se je potrebno izogniti dvakratnemu štetju istih transakcij.

3.4.3 Opredelitev entitete

Entiteta predstavlja naslednjo sestavino, s katero lahko merimo velikost oz. zahtevnost primera uporabe. Z njo se predstavijo tisti sklopi podatkov, ki jih trajno shranijo v zbirki podatkov. Robiola in Orosco predlagata, da se za prepoznavanje števila entitet prav tako uporabi metoda stavčnega razčlenjevanja. Iz besedila PU je potrebno najprej izluščiti vse samostalnike, nato pa iz te množice odstraniti vse, ki niso neposredno povezani s poslovnimi opravili, postopki ali procesi, ki jih želimo podpreti s programsko opremo ([35] 2008; 35). Kot primer lahko navedemo, da so entitete v primeru spletne trgovine lahko izdelek, košarica, kategorija izdelka, račun ..., med entitete pa v tem primeru ne uvrščamo samostalnikov »gumb«, »zaslon«, »brskalnik«, »spustni seznam« itd.

Pri štetju entitet v besedilu je potrebno paziti, da se v primeru večkratnega pojavljanja iste entitete slednja šteje zgolj enkrat. To pravilo izhaja iz predpostavke avtorjev, da je velikost primera uporabe (oz. delovanja programske opreme) odvisna od števila uporabe raznolikih entitet, ne glede na to, kolikokrat se posamezna entiteta pojavi v besedilu. Enako pravilo se uporabi tudi pri zbirki (*angl. module*) večjih primerov uporabe. S pomočjo omenjenih pravil se lahko na podlagi entitet obseg (velikost) delovanja programske opreme izračuna po spodnji enačbi:

$$Obseg\ delovanja = \sum Število\ entitet\ v\ primeru\ uporabe$$

Enačba 24: Izračun obsega delovanja programske opreme s pomočjo števila entitet

Robiola in Orosco navajata ([35] 2008; 40), da lahko obe sestavini, tako transakcijo kot entiteto, uporabijo za določitev velikosti programske opreme in s tem tudi pripravo ocene potrebnega časa, vendar je njuna raziskava pokazala, da je s sestavino »transakcija« mogoče pripraviti oceno z manjšo velikostjo (oz. stopnjo) napake. Kot razlog za to navajata podatek, da so bili v njuno raziskavo vključeni projekti, pri katerih so razvijali programsko opremo za podporo izvedbi poslovnega opravila (oz. procesa), manj pa programska oprema za podporo obdelavi podatkov in iskanju zakonitosti v njih. Za slednje pravita, da gre za podatkovno usmerjeno programsko opremo in manj »funkcijsko« usmerjeno. V okviru praktičnega dela te naloge bodo v raziskavo vključeni projekti, na katerih smo razvijali programsko opremo za podporo izvajanju poslovnih procesov in opravil. Zaradi zgoraj omenjenih ugotovitev se bomo v tem delu posvetili merjenju obsega delovanja programske opreme zgolj s sestavino »transakcija«.

3.4.4 Opredelitev enote za merjenje napake ocene

Produktivnost (*angl. productivity value*) predstavlja osnovno enoto za oceno potrebnega časa v delu Robiole in Orosca ([35] 2008; 36). Izračunamo jo kot količnik med velikostjo (oz. obsegom) programske opreme in porabljenim časom (oz. vloženim trudom) za razvoj programske opreme (glejte enačbo 25). Ocena potrebnega časa za nov projekt se nato izračuna tako, da število vseh preštetih transakcij nove programske opreme delimo s produktivnostjo podobnega preteklega projekta. Avtorja sta oceno potrebnega časa izboljšala še tako, da sta uporabila povprečno produktivnosti vseh podobnih projektov iz preteklosti (2008; 38–39). Produktivnost oziroma povprečna produktivnosti iz drugih projektov bo predstavljala osnovno enoto za pripravo ocene potrebnega časa za naslednje projekte tudi v okviru tega dela.

$$Produktivnost = \frac{velikost\ \ programske\ opreme}{vložen\ trud}$$

Enačba 25: Izračun produktivnosti pri razvoju programske opreme

Ob pripravi ocene moramo razmišljati tudi o stopnji napake, ki se lahko pojavi zaradi pomanjkljivih informacij. Robiola in Orosco v svojem delu omenjata (2008; 34) mero stopnja (oz. razpon) relativne napaka oz. krajše MRE (*angl. magnitude of relative error*), s katero sta določila natačnost ocene. Opredelita jo kot absolutno vrednost količnika med razliko predvidenega in dejanskega vložka z dejanskim vložkom (glejte enačbo 26). Enako mero v svojih delih uporabijo tudi Menzies in drugi ([27] 2006), Jørgensen in Moløkken-Østvold ([18] 2004), Pow-Sang in Jolay-Vasaquez ([34] 2006), Heričko in Živkovič ([12] 2008) ter drugi. Nekateri od teh avtorjev omenjajo tudi mero povprečna stopnja relativne napake oz. krajše MMRE (*angl. mean magnitude of relative error*). Slednjo v nadaljevanju ne bomo uporabljali, saj presega obseg tega dela.

$$Stopnja\ relativne\ napake = \frac{|predvideni\ vložek - dejanski\ vložek|}{dejanski\ vložek}$$

Enačba 26: Izračun stopnje relativne napake ocenjenega časa

Opredelili smo enoto za oceno potrebnega časa in mero za merjenje stopnje relativne napake, s katero bomo določili, za koliko smo se v oceni zmotili. Vendar nam slednji ne ponujata neposrednega odgovora na vprašanje, ali je izbrana metoda ustrezna oz. ali je dovolj natančna, da lahko z njo učinkovito in uspešno pripravljamo ocene potrebnega časa. Odgovor na slednje Robiola in Orosco podata s preprosto enačbo, katere končni izid je odstotek vseh ocen časa (oz. projektov), katerih stopnja povprečne napake je manjša od vnaprej določenega odstotka (glejte enačbo 27). Mero imenujeta »kakovosti priprave ocene«. Avtorja navajata ([35] 2008; 34), da je metoda sprejemljiva oz. uspešna, če se napaka, ki je enaka ali manjša od 25 odstotkov dejansko porabljenega časa (oz. je v meji sprejemljivosti), pojavlja pri vsaj ali več kot 75 odstotkih projektov. Izid enačbe so lahko vse vrednosti med 0 in 1. Vrednost 1 pomeni, da je pri vseh projektih vrednost napake manjša ali enaka meji sprejemljivosti, vrednost 0,75 pa npr. pomeni, da je napaka v okviru meje le pri 75 odstotkih projektov. Nekateri avtorji ([27] Menzies in drugi, 2006; 886) postavljajo mejo sprejemljivosti tudi na 30 odstotkov

napake. V okviru te naloge bomo uporabili strožjo mejo – 25 odstotkov – in tako poskušali doseči zastavljeni tretji cilj.

$$\frac{\text{Kakovost priprave}}{\text{ocene (meja sprejemljivosti)}} = \frac{\text{št. projektov v okviru meje sprejemljivosti}}{\text{št. vseh projektov}}$$

Enačba 27: Izračun kakovosti metode za pripravo ocene časa

3.4.5 Razlogi za izbiro metode

Z izbiro te metode smo izpolnili prvi cilj te naloge, izidi praktičnega dela pa bodo potrdili ali ovrgli pravilnost izbire. Razlogov za izbiro prav te metode je več. Prvi je prav gotovo lastnost, da lahko s to metodo pripravimo oceno potrebnega časa na zelo preprost in hiter način. Pri metodi funkcijskih točk je potrebno za pripravo ocene uporabiti pet različnih sestavin programske opreme in množico dodatnih 14 dejavnikov, se pravi vsaj 19 različnih sestavin, s čimer se poveča zahtevnost uporabe metode. Slednja tudi ne omogoča časovno hitre priprave ocene, saj glavnih gradnikov programske opreme ni mogoče prešteti prej kot v stopnji načrtovanja programske opreme. Se pravi, za oceno časa je potrebno opraviti zajem zahtev, analiziranje in na koncu še načrtovanje programske opreme, za kar je vsekakor potrebno veliko časa. Metoda točk primerov uporabe to pomanjkljivost odpravi, saj je za oceno časa potrebno opraviti zgolj zajem zahtev in analiziranje. Vendar pa ne odpravi velikega števila dejavnikov oz. sestavin. Pri pripravi ocene je potrebno upoštevati sestavini »vmesnik« in »transakcija« ter 13 tehničnih (glejte preglednico 4) in 8 okoljskih dejavnikov (glejte preglednico 5), skupaj z dodatnim prilagoditvenim faktorjem. Zaradi slednjega metoda ni med preprostejšimi.
Za razliko od navedenih metod pa se izbrana metoda osredotoča zgolj na dve sestavini (oz. dejavnika): transakcijo in entiteto. V našem delu celo zgolj na eno samo – transakcijo. Tukaj je potrebno poudariti, da pri pripravi ocene sploh ne upoštevajo zahtevnosti posamezne transakcije. Ocena je neodvisna od zahtevnosti (oz. zapletenosti) transakcij, odvisna je zgolj od števila transakcij. Metoda temelji na predpostavki, da ima večja in zahtevnejša programska oprema večje število transakcij. Za lažje razumevanje navedimo primer iz avtomobilske industrije: izbrana metoda bi oceno časa za razvoj novega avtomobila dobila na podlagi števila sestavnih delov vozila ali npr. na podlagi števila gibljivih sklopov avtomobila in to neodvisno od tega, ali so deli oz. gibljivi sklopi veliki, majhni, železni ali plastični itd. Mogoče bi se celo lahko izkazalo, da je čas razvoja novega avtomobila z npr. 5000 deli ali 400 gibljivimi sklopi podoben času razvoja novega nebotičnika s podobnim številom delov oz. gibljivih sklopov oz. času razvoja nove organizacije združbe s podobnim številom sestavnih delov. Pomembno je zgolj število določenih sestavnih delov. Ravno zaradi tega je metoda preprosta in lahka za uporabo. Ključni razlog je tudi čas nastanka ocene. Pri tej metodi se ocena pripravi v zgodnji stopnji projekta, v času analiziranja ali celo prej. O tem je bilo že veliko napisanega v predhodnih poglavjih, zato se na tem mestu ne bomo spuščali v podrobnosti.

Naslednji razlog za uporabo metode je njena prilagodljivost. Čeprav Robiola in Orosco natančno opredelita transakcijo, lahko vsak uporabnik te metode uporabi za njo svojo opredelitev. Pomembno je zgolj to, da je pri opredelitvi in uporabi opredelitve skladen in dosleden. Prilagodljivost metode se izkazuje tudi v dejstvu, da lahko transakcije zaznamo in preštejemo tudi v drugih listinah, ki niso nujno listine »primeri uporabe«. Številne metode

razvoja programske opreme uporabljajo za opis obsega in delovanja programske opreme svoje vrste listin. Primeri teh listin so:

- opis zgodb (*angl. stories*) v prilagodljivejših metodologijah (*angl. agile methodology*),
- opisi storitev v storitveno usmerjeni metodologiji, krajše SOMA (*angl. service-oriented modeling and architecture*),
- diagrami tokov podatkov (*angl. dataflow diagrams*) v okviru metode SSDAM (angl. *structured-systems analysis and design method*) itd.

Pri pregledu omenjenih listin je potrebno prepoznati in prešteti posamezne transakcije. Mogoče v tem primeru termin »transakcija« sploh ni več ustrezen, ampak bi bila ustreznejša besedna zveza »informacijsko podprto poslovno opravilo«. Sodbo o pravilnosti avtorjevega razmisleka prepuščam bralcu.

Pomemben del prilagodljivosti metode predstavlja neprestano izboljševanje ocenjevanja na podlagi zbranih podatkov iz že izvedenih projektov. To v svojem delu poudarjata tudi Robiola in Orosco ([35] 2008; 34). Glede na opravljeno raziskavo ([35] Robiola in Orosco, 2008) daje metoda tudi zelo spodbudne učinke, saj je ocenjevanje na podlagi transakcij ponudilo ocene, katerih napaka je bila manjša od 25 odstotkov. Če bomo pri praktičnem delu te naloge dobili enake izide, bomo dosegli zastavljeni tretji cilj te naloge. Slednje predstavlja dodaten razlog za izbiro te metode.

Osnovni namen te naloge je preučiti delo Robiole in Orosca in opraviti novo raziskavo, ki bo nadgradila njune ugotovitve tako, da bo ponudila nova spoznanja in hkrati čim bolj potrdila njune predhodne učinke. Prav zaradi tega se nova raziskava oz. praktični del razlikuje v naslednjih stvareh:

- Ocene produktivnosti ne bomo pripravili za izvedbo celotnega projekta, ampak zgolj za dve stopnji razvoja programske opreme, kot sta analiziranje in načrtovanje. Če bi obstajale ocene tudi za ostale stopnje, bi končno oceno predstavljala vsota vseh posameznih ocen.
- Opredelitev transakcije se bo razlikovala od opredelitve, ki sta jo podala Robiola in Orosco. Slednje je potrebno zato, ker primerov uporabe v nekaterih projektih, ki bodo vključeni v praktični del, ni bilo. Za opis obsega delovanja so bili na voljo podatki iz drugačnih listin. Zaradi tega tudi ni bilo mogoče narediti stavčne razčlenitve, kot jo predlagata Robiola in Orosco.
- Robiola in Orosco sta za vsak projekt na podlagi zbranih podatkov najprej izračunala produktivnost, nato pa zanj še povprečno produktivnost ([35] 2008; 38–39) na podlagi ostalih projektov. To je bilo mogoče šele po tem, ko so bili vsi projekti že končani. Iz tega lahko sklepamo, da sta avtorja pripravila ocene šele po že izvedenih projektih. V okviru tega dela bomo tudi sproti izračunali povprečno produktivnost. Pri prvem projektu povprečna produktivnost še ne bo obstajala, pri drugem projektu jo bomo izračunali na podlagi prvega projekta, pri tretjem projektu na podlagi prvih dveh in tako naprej. Na koncu bomo naredili tudi enako obdelavo podatkov, kot sta jo naredila Robiola in Orosco. Tako bomo lahko primerjali in potrdili pravilnost raziskave teh dveh avtorjev.

4 Ugotovitve raziskave za metodo poenostavljenih točk primerov uporabe

4.1 Opis raziskave in opis poteka izvedbe raziskave

4.1.1 Predstavitev raziskave

Raziskavo sem želel izvesti pod čim bolj stvarnimi pogoji. Zato sem se odločil, da raziskavo izvedem v okviru okolja združbe, za katero sem delal od jeseni leta 2004 do začetka leta 2009. Ta združba je na področju računalniške dejavnosti ena izmed največjih zasebnih združb v Sloveniji. Zaposluje več kot 400 ljudi, za naročnike pa gradi programsko opremo na različnih tehnologijah, kot so npr. Java, .NET, Delphi, CICS, Domino, Oracle zbirke podatkov, IBM zbirke podatkov in še bi lahko naštevali. Njene največje stranke so stranke iz državne uprave, bančništva, področja telekomunikacij itd.

V omenjeni združbi sem delal ves čas v okviru projektne skupine, ki je bila po sestavi dokaj stalna. Sestavljali so jo: ravnatelj projekta, eden do dva analitika, načrtovalec, eden do dva preizkuševalca (oz. »testerja«), tri do štirje razvijalci na področju zbirk podatkov in šest do sedem Java razvijalcev. Ta projektna skupina je med leti 2004 in 2008 uspešno izvedla kar nekaj različno velikih projektov, pretežno na področju javne uprave. V to raziskavo sem vključil pet izmed teh projektov. Ker so imena teh projektov in zbrani podatki poslovna skrivnost podjetja, sem projekte poimenoval z navideznimi, preprostimi imeni. V raziskavo so bili vključeni naslednji projekti:

- Projekt A1: izgradnja spletnega sistema za poslovanje z državljani.
- Projekt A2: izgradnja spletnega sistema za poslovanje z državljani.
- Projekt B1: nadgradnja obstoječega, informacijsko podprtega, državnega registra z novo evidenco.
- Projekt C1: izgradnja novega, informacijsko podprtega, državnega registra.
- Projekt C2: izgradnja novega, informacijsko podprtega, državnega registra.

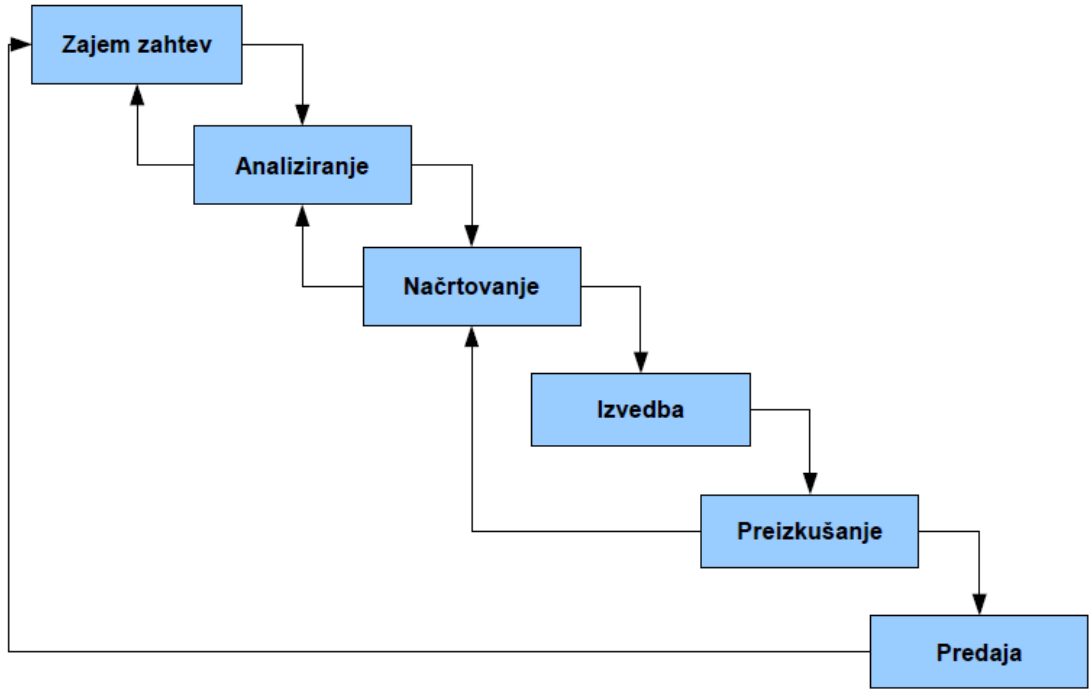
V okviru teh petih projektov sem opravljal naloge poslovnega analika, načrtovalca in preizkuševalca (oz. testerja) programske opreme. Kot poslovni analitik sem skrbel za komuniciranje z naročnikom, zbiranje zahtev naročnika in urejanje ter popis zahtev v listinah »primeri uporabe«. Kot načrtovalec sem kot vhodne listine prejel listine »primeri uporabe«, nato pa sem zahteve v listinah preoblikoval v sistematično urejeno, nedvoumno sestavo oz. v vsebinski načrt programske opreme. Ta sestava je bila zapisana v obliki logičnega modela zbirke podatkov in obliki postopkov (oz. algoritmov), ki so opisovale delovanje transakcij programske opreme. Kot preizkuševalec sem opravljal preizkuse vsebine in zmogljivosti programske opreme. Vhodne listine za ta del so bile tudi listine »primeri uporabe«.

Na projektu je bil vsak zaposlenec dolžan beležiti čas, ki ga je porabil za opravljanje določene delovne naloge. V okviru projekta ni bilo opredeljeno, kako natančno je treba beležiti porabljen čas, ob tem pa ni potekal ustrezen nadzor, ki bi zagotavljal ustrezno kakovost zabeleženih podatkov. Dostop do teh podatkov so imeli zgolj ravnatelj projekta, ravnatelj programa (omenjeni projekti so bili v okviru njegovega stroškovnega mesta) in izvršni ravnatelj. Orodje, v katerega smo beležili čas, ni omogočalo vpogleda in analize (oz. razčlenitve) lastnega porabljenega časa. Zaradi omenjenih razlogov sem se odločil, da bom za namen te raziskave uporabil le svoj zabeležen porabljen čas za posamezni aktivnosti, kot sta

analiziranje in načrtovanje. V ta namen sem porabljen čas vnašal v lastno evidenco, za vsako delovno nalogo na 15 minut natančno. Več o tem v nadaljevanju.

4.1.2 Predstavitev okolja, v katerem je bila izvedena raziskava

Omenjena združba se ukvarja z računalniško dejavnostjo že več kot 20 let. V tem času je uspela razviti nekaj lastnih metod za razvoj programske opreme. Te metode je združba pogosto prilagodila oz. osnovala za potrebe določenih sklopov projektov in naročnikov na podlagi splošne, javno dostopne metode. Podrobnosti o prilagajanju metod razvoja programske opreme za potrebe določene združbe si lahko več preberete v delu avtorjev Bajec, Vavpotič in Krisper ([2] 2007). Med prilagojenimi metodami je združba razvila je tudi metodo, ki temelji na zaporednem in padajočem (oz. stopničastem) procesnem modelu (glejte sliko 10). To metodo smo uporabili pri uresnišitvi vseh petih projektov, vključenih v raziskavo.



Slika 10: Prikaz stopenj procesa razvoja programske opreme, v okviru katerega je potekala raziskava

Pri tej metodi stopnje (oz. aktivnosti) projekta izvajajo zaporedno, z možnostjo vrnitve in ponovne izvedbe ene izmed predhodno izvedenih aktivnosti. Metoda predvideva naslednje stopnje razvoja:

1) **Zajem zahtev**: V okviru te stopnje naročnik preko sestankov, listin in pogovorov poda svoje vsebinske in nevsebinske zahteve glede na želeno programsko opremo. Po sprejemu zahtev nato izvajalec slednje pregleda in preveri, ali je razumevanje zahtev skladno z naročnikovim razumevanjem.
2) **Analiziranje** (oz. proučevanje): Po opravljenem zajemu zahtev je vsebinske zahteve potrebno podrobneje razčleniti in na podlagi zahtev opredeliti uporabniški vmesnik programske opreme. Sestavni del te stopnje je tudi natančna opredelitev vseh funkcij, organizacijskih predpisov ali funkcionalnih in nefunkcionalnih pravil ([28] Mihelčič,

2010; 24), katere je potrebno podpreti z zahtevano programsko opremo. Za nevsebinske zahteve je potrebno podrobneje opredeliti arhitekturne, varnostne, zmogljivostne, strojne in druge zahteve. Na podlagi natančnejših opredelitev zahtev izdelajo predlog rešitve, sestavljen iz seznama listin »primeri uporabe« (v nadaljevanju PU) in osnutka modela zbirke podatkov. Listino PU lahko primerjamo s tehnološkimi listinami v okviru tehnološke priprave proizvodnje v proizvajalnih podjetjih ([31] Mihelčič, 2004; 197). Tako kot tehnološke listine tudi PU opisujejo sestavo delovanja programske opreme. Vsak PU podrobneje razdela samostojni logični sklop zahtev. Poenostavljeno bi lahko rekli, da PU vsebujejo natačne opredelitve poslovnih opravil, ki jih je potrebno informacijsko podpreti s programsko opremo.

3) **Načrtovanje**: V tej stopnji a podlagi listin PU izdelajo vsebinske načrte transakcij oz. informacijsko podprtih poslovnih opravil. Vsebinski načrti so listine, v katerih so transakcije popisane v nedvoumni obliki, v obliki UML (*Unified Modeling Language*) diagramov in v obliki navidezne kode (oz. psevdo kode). V okviru transakcij so nedvoumno opredeljena tudi pravila za kontrolo vnosa podatkov na uporabniškem vmesniku in pri povezovanju z zunanjimi sistemi, organizacijski predpisi ter nefunkcionalna pravila. Načrtovalec je zadolžen za izdelavo podrobnega logičnega modela zbirke podatkov. V okviru te stopnje izdelajo tudi načrte za nevsebinske zahteve glede tehnološke arhitekture programske opreme, postavitve strojne opreme, varnostne arhitekture programske opreme itd.
4) **Izvedba** (oz. implementacija): V tej stopnji razvijalci in sistemski inženirji na podlagi PU in načrtov iz predhodne stopnje postavijo strojno opremo ter s pomočjo predhodno izbrane tehnologije izdelajo programsko opremo, skladno z vsebinskimi in nevsebinskimi zahtevami.
5) **Preizkušanje** (oz. testiranje): Na podlagi PU v listinah za preizkušanje opredelijo in popišejo postopke (oz. scenarije) za preizkušanje programske opreme. Poenostavljeno povedano, vsak postopek omogoča izvedbo preizkusa »obnašanja« informacijsko podprtega poslovnega opravila v določenih okoliščinah. Preizkuševalec nato s pomočjo listin za preizkušanje izvede ustrezen nabor preizkusov in si zabeleži podatke o morebitnih napakah. Te podatke nato posreduje analitiku, načrtovalcu in/ali razvijalcu z namenom odprave najdenih napak.
6) **Predaja**: po uspešno opravljenem preizkusu pri naročniku in prevzemu ustreznega nabora listin lahko izvedejo predajo programske opreme. Po predaji lahko izdelano programsko opremo uporabijo v vsakdanjem poslovanju naročnika.

Delo na predhodno omenjenih petih projektih je pokazalo, da lahko podatke iz PU uporabimo v različnih aktivnostih na projektu. Napisali smo že, da PU uporabljamo kot vhodne listine v stopnjo načrtovanja in v stopnjo preizkušanja za pripravo in popis postopkov preizkušanja. Ob odkriti napaki lahko s PU hitreje ugotovimo, kje v programski opremi se napaka nahaja, in to pogosto brez potrebe po sodelovanju z razvijalcem programske opreme. Hkrati lahko PU uporabimo tudi za pripravo uporabniških navodil in listin za pomoč pri izvajanju podpore končnim uporabnikom programske opreme. Pri novi zaposlitvi ali prerazporeditvi zaposlenca združbe lahko PU uporabimo za hitrejše in bolj kakovostno uvajanje zaposlenca pri delu na projektu. Iz tega lahko tudi sklepamo, da je v PU trajno shranjeno znanje o vsebini projektov, kar je za samo združbo dandanes zelo pomembno. In na koncu, kot smo v tem delu že velikokrat omenjali, lahko PU uporabimo tudi za pripravo ocene potrebnega časa za razvoj programske opreme ali pripravo ocene potrebnega časa za posamezne stopnje razvoja programske opreme. Slednje bomo poskušali dokazati v nadaljevanju.

4.1.3 Opis projektov in dejavnikov projektov, ki so vplivali na raziskavo

Kot sem že napisal, je bilo v raziskavo vključenih pet projektov s področja javne uprave. Uresničevanje projektov je potekalo zaporedno, projekt po projektu. Najprej je bil uspešno izpeljan projekt A1, nato projekt A2, sledil mu je projekt B1, slednjemu pa projekta C1 in C2. Zaporedje uresničevanja projektov je delno vplivalo tudi na pripravo ocen potrebnega časa, saj je bila, kot bomo videli kasneje, v nekaterih primerih ocena potrebnega časa odvisna zgolj od podatkov o porabljenem času že izvedenih projektov.

Za uspešno izvedbo raziskave je bilo potrebno projekte opisati in razvrstiti tako, da smo lahko zbrane podatke o ocenah porabljenega časa in podatke o dejansko porabljenem času med seboj primerjali, preučili ter zbrane podatke uporabili za pripravo oceno porabljenega časa na naslednjem projektu. Ker smo v okviru raziskave ocenjevali potreben čas zgolj za posamezne stopnje razvoja programske opreme, smo pri opisu in razvrstitvi projektov uporabljali le tiste dejavnike, od katerih so posamezne stopnje odvisne. Tako, na primer, dejavniki, kot so uporabljena tehnologija ali izkušnje razvijalcev, niso imeli večjega pomena v okviru raziskave, saj je od njih odvisna predvsem stopnja izvedbe (oz. implementacije) programske opreme, za katero pa nismo pripravljali ocene potrebnega časa.

V okviru raziskave smo pri ocenjevanju potrebnega časa za stopnjo analiziranja in načrtovanje vključili naslednja dejavnika:

- vsebina projekta ter
- obseg projektne skupine. Na nekaterih projektih je bilo potrebno za uspešno uresničitev projekta združiti delo več projektnih skupin, pri nekaterih projektih pa je bilo za uresničitev dovolj le delo osnovne projektne skupine.

Pri ocenjevanje potrebnega časa za stopnjo analiziranja smo vključili dodatne dejavnike, kot so:

- izkušnje naročnika s predhodnimi, podobnimi projekti,
- oblika in urejenost sestave vhodnih listin v stopnji analiziranja. Vhodne listine so lahko: zakoni, okvirni popis zahtev naročnika, zapisniki sestankov, primerki listin, ki jih sedaj uporabljajo v poslovnih procesih, itd.
- Oblika in urejenost sestave izhodnih listin v stopnji analiziranja. Izhodne listine so lahko: PUji, model zasnove zbirke podatkov, ekranske slike programske opreme, podroben popis funkcij programske opreme v nestandardni obliki in sestavi itd.
- Izkušenost analitika (oz. analitikov), ki je izvedel stopnjo analiziranja.

Ocenjevanje potrebnega časa za stopnjo načrtovanja je bilo odvisno še od naslednjih dodatnih dejavnikov:

- oblike in urejenosti sestave vhodnih listin v stopnji načrtovanja. Vhodne listine so lahko: PU, model zasnove zbirke podatkov, ekranske slike programske opreme, podroben popis funkcij programske opreme v nestandardni obliki in sestavi, opis arhitekture sistema, opis varnostne arhitekture itd.;
- oblike in urejenosti sestave izhodnih listin iz stopnje načrtovanja. Izhodne listine so lahko: vsebinski načrti transakcij in logični, vsebinski model zbirke podatkov;
- uporabljene tehnologije za razvoj programske opreme;
- izkušenosti načrtovalca, ki je izvedel stopnjo načrtovanja;

Za učinkovito analizo pridobljenih podatkov raziskave je bilo potrebno vsakemu dejavniku opredeliti zalogo vrednosti. Na ta način sem lahko v okviru analize izidov raziskave primerjal ocene potrebnega časa glede na dejavnike, v okviru katerih smo uresničili projekte. Zaloge vrednosti so opredeljene v preglednici 6:

Stopnja razvoja prog. opreme	Dejavnik	Zaloga vrednosti
Analiziranje	Izkušnje naročnika	• Izkušen
		• Neizkušen
	Oblika in urejenost sestave vhodnih listin	• Neurejena oblika in sestava
		• Urejena oblika in sestava
	Oblika in urejenost sestave izhodnih listin	• Neurejena oblika in sestava
		• Urejena oblika in sestava
	Izkušenost analitika	• Izkušen
		• Neizkušen
	Vsebina projekta	• Spletna aplikacija
		• Register
	Obseg projektne skupine	• Osnovna
		• Razširjena
Načrtovanje	Oblika in urejenost sestave vhodnih listin	• Neurejena oblika in sestava
		• Urejena oblika in sestava
	Oblika in urejenost sestave izhodnih listin	• Neurejena oblika in sestava
		• Urejena oblika in sestava
	Uporabljene tehnologije	• Javanski strežnik IBM in zbirka podatkov Oracle
		• Javanski strežnik IBM, javanski strežnik Oracle in zbirka podatkov Oracle
		• Javanski strežnik IBM, javanski strežnik Oracle, strežnik IBM Domino, zbirka podatkov IBM Domino in zbirka podatkov Oracle
	Izkušenost načrtovalca	• Izkušen
		• Neizkušen
	Vsebina projekta	• Spletna aplikacija
		• Register
	Obseg projektne skupine	• Osnovna
		• Razširjena

Preglednica 6: Zaloge vrednosti posameznih dejavnikov projektov razvoja programske opreme

Na preglednici 7 so opisani dejavniki, ki so bili prisotni ob uresničitvi posameznih projektov.

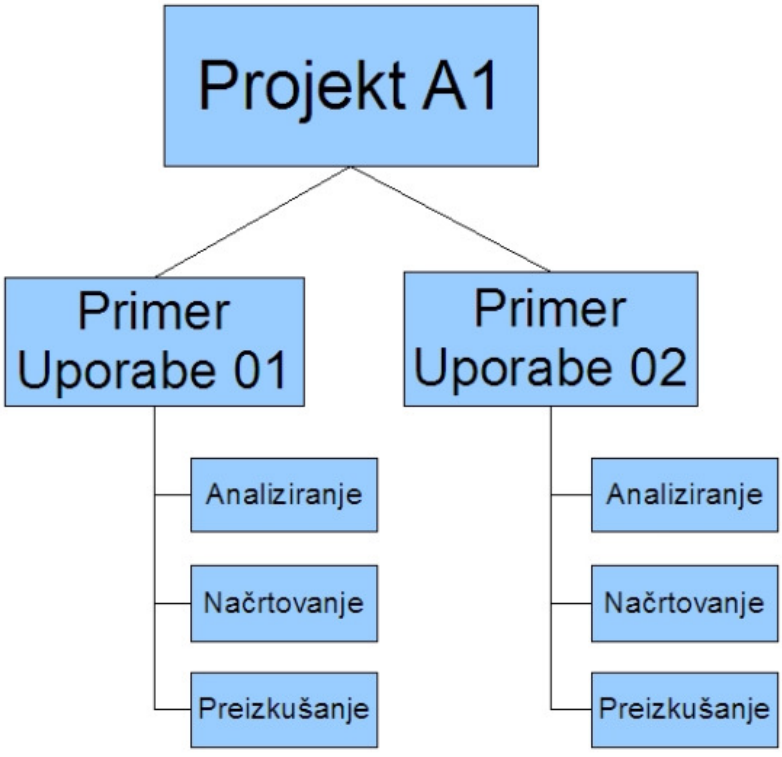
Stopnja razvoja prog. opreme	Dejavnik	Projekt				
		A1	A2	B1	C1	C2
Analiziranje	Izkušnje naročnika	Neizkušen	Neizkušen	Izkušen	Izkušen	Izkušen
	Oblika in urejenost sestave vhodnih listin	Neurejena	Neurejena	Urejena	Urejena	Urejena
	Oblika in urejenost sestave izhodnih listin	Neurejena	Neurejena	Urejena	Urejena	Urejena
	Izkušenost analitika	Neizkušen	Neizkušen	Izkušen	Izkušen	Izkušen
	Vsebina projekta	Spletna aplikacija	Spletna aplikacija	Register	Register	Register
	Obseg projektne skupine	Razširjena	Razširjena	Osnovna	Osnovna	Osnovna
Načrtovanje	Oblika in urejenost sestave vhodnih listin	Neurejena	Neurejena	Urejena	Urejena	Urejena
	Oblika in urejenost sestave izhodnih listin	Neurejena	Neurejena	Neurejena	Urejena	Urejena
	Uporabljene tehnologije	Javanski strežnik IBM, javanski strežnik Oracle in zbirka podatkov Oracle	Javanski strežnik IBM, javanski strežnik Oracle, strežnik IBM Domino, zbirka podatkov IBM Domino in zbirka podatkov Oracle	Javanski strežnik IBM in zbirka podatkov Oracle	Java strežnik IBM in zbirka podatkov Oracle	Javanski strežnik IBM in zbirka podatkov Oracle
	Izkušenost načrtovalca	Neizkušen	Neizkušen	Neizkušen	Izkušen	Izkušen
	Vsebina projekta	Spletna aplikacija	Spletna aplikacija	Register	Register	Register
	Obseg projektne skupine	Razširjena	Razširjena	Osnovna	Osnovna	Osnovna

Preglednica 7: Opis dejavnikov po posameznih projektih

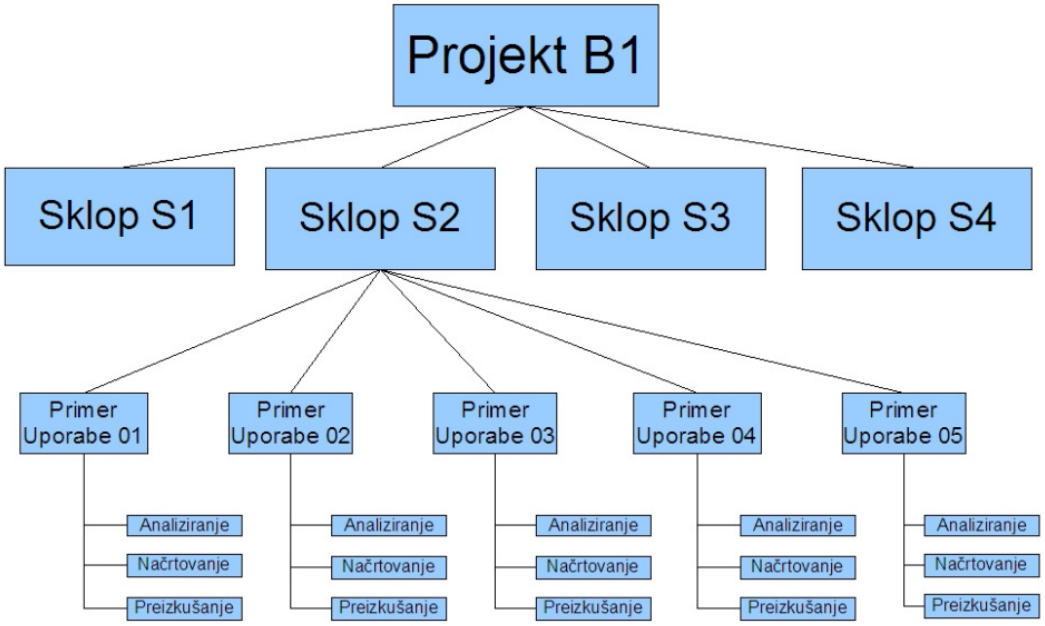
4.1.4 Predstavitev metod dela

V raziskavi smo za izbrane projekte najprej izdelali podroben načrt dela oz. sestavo WBS, na podlagi katere smo lahko kasneje pripravili oceno potrebnega časa. Z WBS smo pravzaprav opisali sestavo programske opreme, iz katere je nato razviden načrt dela. Sestavo programske opreme smo členili do ravni listin »primeri uporabe« (v nadaljevanju PU). Vsak PU opisuje eno ali več poslovnih opravil, vsako poslovno opravilo pa predstavlja osnovno enoto, katero je potrebno informacijsko podpreti. Tej osnovni enoti lahko rečemo tudi transakcija. Na ravni PU se lahko uporabljajo tudi listine, ki opisujejo splošno, logično zaokroženo delovanje programske opreme, skupno večjemu številu PU. Te listine vsebujejo opise skupnega modela zbirke podatkov, skupnih organizacijskih predpisov ali funkcionalnih ter nefunkcionalnih pravil, skupne ekranske slike itd.

Izdelani WBS so podobni sestavi delovanja programske opreme (glejte sliko 7). Razlikujejo se le v tem, da so na končnih gradnikih (oz. listih), takoj za ravnijo PU-jev, navedene stopnje oz. aktivnosti procesa razvoja programske opreme, ki jih je potrebno opraviti za določen sklop delovanja oz. za določeno poslovno opravilo. Za lažje razumevanje sta na naslednjih slikah prikazani poenostavljeni sestavi WBS v obliki drevesa za projekt A1 in projekt B1. Zaradi varovanja poslovne skrivnosti je v sestavi WBS navedeneno navidezno število gradnikov z navideznimi imeni. Ker za namen raziskave potrebujemo zgolj število transakcij (oz. informacijsko podprtih poslovnih opravil), bom prikaz sestave WBS za ostale projekte izpustil. V nadaljevanju pa bom s pomočjo preglednice predstavil stvarne podatke o številu transakcij in porabljenem številu ur po posameznih PU in aktivnostih za vsak projekt posebej.



Slika 11: Primer sestave WBS za projekt A1



Slika 12: Primer sestave WBS za projekt B1

Robiola in Orosco ([35] 2008; 35) v svojem delu navajata nabor pravil, s katerimi lahko v PU-jih na enostaven način preštejemo število transakcij. Njuna osnovna zamisel temelji na uporabi slovničnega stavčnega razčlenjevanja. V okviru te raziskave se ne bomo poslužili tega načina, ampak bomo kot transakcije opredelili samostojne logične sklope (oz. funkcije), ki jih lahko izvedemo s programsko opremo. Te sklope si lahko predstavljamo tudi kot poslovna opravila, ki jih je potrebno informacijsko podpreti, ali tudi kot spletne storitve, ki jih bo programska oprema ponujala zunanjemu okolju. Za lažje razumevanje bomo k vsaki vrsti transakcije navedli primer iz spletnega sistema za podporo delu z elektronsko pošto. Vrste transakcij (oz. logičnih sklopov) so naslednje:

- **samostojno iskanje ali pregled seznama entitet ali sklopa entitet:**
  Kot primer te vrste transakcije lahko navedem funkcijo prikaza prispele elektronske pošte. Slednja se izvede ob prijavi v sistem za elektronsko pošto ali na zahtevo uporabnika; ob kliku na povezavo (ali gumb) »Prispela pošta«. Ob izvedbi te funkcije se najprej poišče vsa elektronska pošta za prijavljenega uporabnika, zatem pa se seznam najdene elektronske pošte tudi prikaže.
- **samostojno iskanje ali pregled podrobnosti posamezne entitete ali sklopa entitet:**
  Ko želimo pogledati vsebino nove elektronske pošte, kliknemo na neprebrano sporočilo. V tem primeru se izvede iskanje podrobnih podatkov neprebranega sporočila in prikaz le-teh. Drugi možen primer te vrste transakcije bi lahko bil pregled podrobnih podatkov o osebi, katere imamo shranjene v imeniku našega spletnega sistema.
- **samostojno dodajanje entitete ali sklopa entitet preko uporabniškega vmesnika:**
  Pošiljanje novega sporočila ali dodajanje nove osebe v imenik predstavlja primer te vrste transakcije.
  Če gradimo (oz. prenavljamo) nov informacijski sistem in želimo vanj iz obstoječega informacijskega sistema prenesti stare podatke, lahko v ta sklop uvrščamo tudi transakcije, ki so zadolžene za omenjeni prenos podatkov. Takšen primer bi lahko bil

uvoz podatkov o osebah iz datoteke, ki smo jo pripravili v drugem sistemu za delo z elektronsko pošto.

- **samostojno spreminjanje entitete ali sklopa entitet:**
  Kot primer te vrste transakcije lahko navedem spreminjanje podatkov o osebi v imeniku. Če bi lahko že poslano elektronsko pošto spremenili in jo ponovno poslali, bi lahko slednjo transakcijo šteli kot primer te vrste. Kot dodaten primer lahko navedemo samodejni prenos elektronske pošte v mapo za nezaželeno pošto. V tem primeru sistem spremeni status prejete pošte v »nezaželeno«.
- **samostojno odstranjevanje entitete ali sklopa entitet:**
  Tipičen primer odstranjevanja je brisanje elektronske pošte iz »koša« (*angl. trash*). Primer te vrste je tudi brisanje osebe iz imenika.
- **samostojni izvoz podatkov entitet ali sklopa entitet v obliki elektronskih listin:**
  Primer te vrste transakcije je izvoz podatkov o osebah iz imenika v datoteko.
  Ta vrsta transakcije je le ena izmed oblik iskanja in pregleda entitet, pri čemer se podatki prikažejo (oz. izvozijo) v posebno elektronsko listimo (npr. PDF, XML, datoteko tipa .doc itd.). Ker se na projektih, vključenih v raziskavo, velikokrat pojavlja omenjeni izvoz podatkov, sem omenjeni izvoz opredelil kot novo vrsto transakcije.

Vsakič, ko v PU zaznamo samostojni logični sklop, ki ga lahko umestimo v eno izmed zgornjih vrst transakcij, lahko sklepamo na prisotnost ene transakcije. Če imamo npr. v PU navedene štiri vrste transakcij (oz. logične sklope), pomeni, da imamo v PU štiri transakcije. To pravilo ne velja za logične sklope, pri katerih se sklicujemo na logični sklop drugega PU. V tem primeru takšno sklicevanje na logični sklop ne obravnavamo kot novo, dodatno transakcijo. Do vseh zgoraj navedenih logičnih sklopov (oz. transakcij) lahko uporabnik dostopa preko uporabniškega vmesnika (kot človek) ali preko elektronskega vmesnika – spletne storitve (kot zunanji informacijski sistem). Če lahko do transakcije dostopa po obeh poteh, se transakcija šteje dvakrat.

V okviru raziskave smo ustvarili lastno evidenco porabljenega časa, v katero smo beležili podatke o porabljenem času. Na podlagi te evidence smo lahko po končanem zbiranju izvedli ustrezno obdelavo zbranih podatkov in analizo dobljenih izidov. Evidenca je bila shranjena v datoteki programa Microsoft Excel, v njej pa smo beležili naslednje podatke (preglednica 8):

- datum: delovni dan,
- čas: čas začetka opravljanja določene delovne naloge,
- obdobje: porabljen čas za opravljanje določene delovne naloge,
- projekt: projekt, na katerem je bilo opravljena delovna naloga,
- sklop: sklop (ali tudi določen PU) projekta, za katerega je bilo opravljeno delo,
- aktivnost: posamezna aktivnost oziroma stopnja razvoja programske opreme, opravljena za določen sklop projekta,
- opis: prosti opis opravljene delovne naloge.

Podatke smo med delom sproti zapisovali v datoteko (glejte preglednico 8). Za določitev sklopa in aktivnosti v posameznem projektu smo uporabil sestavo WBS tega projekta. Če izbira sklopa ali aktivnosti za določeno nalogo ni bi bila jasno določena, smo slednjo določili skupaj z ravnateljem projekta in vodilnim analitikom.

Datum	Čas	Obdobje	Projekt	Sklop	Aktivnost	Opis delovne naloge
Skupaj		**8:10**				
29.10.2008	7:00	0:30	Podporni projekt	-	Administracija	Branje elektronske pošte, administracija
29.10.2008	7:30	1:45	Projekt C2	Primer uporabe 02	Načrtovanje	Priprava načrta za PU02
29.10.2008	9:15	0:30	Podporni projekt	Mentorstvo osebe A1	Mentorstvo	Z osebo A pregledal vprašanja in logični model
29.10.2008	9:45	0:15	Projekt C1	Primer uporabe 05	Načrtovanje	Delo povezano z opisom opravil 71 in 140
29.10.2008	10:00	1:15	Projekt C2	Primer uporabe 03	Načrtovanje	Z analitikom pregledal odprte točke pri PU03
29.10.2008	11:15	1:00	Podporni projekt	-	Malica	Malica
29.10.2008	12:15	1:00	Podporni projekt	-	Administracija	Pregled opravil, njihovo združevanje v skupine
29.10.2008	13:15	0:30	Podporni projekt	Mentorstvo osebe A1	Mentorstvo	Pomoč osebi A pri načrtovanju
29.10.2008	13:45	0:15	Projekt C1	Primer uporabe 49	Načrtovanje	Priprava opisa opravila 161
29.10.2008	14:00	0:30	Podporni projekt	-	Raziskave	Priprava na sestanek
29.10.2008	14:30	0:45	Podporni projekt	-	Raziskave	Sestanek
29.10.2008	15:15					
Skupaj		**6:00**				
30.10.2008	7:00	0:30	Podporni projekt	-	Administracija	Branje elektronske pošte, administracija
30.10.2008	7:30	0:45	Projekt C2	Logični model podatkov	Načrtovanje	Pomoč razvijalcem pri izvedbi
30.10.2008	8:15	0:45	Podporni projekt	Mentorstvo osebe A1	Mentorstvo	Osebi A razložil koncept transakcije
30.10.2008	9:00	0:30	Projekt C1	Primer uporabe 03	Načrtovanje	Nadgradnja načrta za PU03
30.10.2008	9:30	2:00	Projekt C1	Logični model podatkov	Načrtovanje	Priprava opisa opravila 96
30.10.2008	11:30	1:00	Podporni projekt	-	Malica	Malica
30.10.2008	12:30	0:30	Projekt C2	Logični model podatkov	Načrtovanje	Priprava opisa opravil za razvijalce
30.10.2008	13:00					

Preglednica 8: Primer zapisov porabljenega časa za posamezen projekt, sklop (oz. primer uporabe) in aktivnost

V okviru izvedbe raziskave smo želeli odstraniti čimveč dejavnikov, ki bi lahko vplivali na kakovost pridobljenih izidov. To smo najlažje dosegli z izbiro takšnih projektov, ki smo jih kot projektna skupina uresničili pod podobnimi okoliščinami in za katere smo lahko zagotovili nepristransko in celovito zbiranje podatkov o porabljenem času za opravljanje naloge. Z izbiro že opisanih petih projektov sem izpolnil drugi cilj tega dela. Projekte sem izbral zato, ker:

- jih je uresničila ista projektna skupina, katere sestava je bila pri vseh projektih bolj ali manj enaka. S tem smo dosegli, da so imeli ljudje na vseh petih projektih enako znanje;
- smo pri vseh projektih razvili programsko opremo za podporo izvajanju poslovnih procesov in opravil;
- smo pri vseh projektih razvijali novo programsko opremo in ni šlo za projekte vzdrževanja programske opreme;
- smo pri vseh projektih uporabili enak proces razvoja programske opreme;
- smo bili pri izbranih projektih zadolženi za izvedbo stopenj analiziranja in načrtovanja za posamezne sklope ali tudi celotno programsko opremo. Na ta način smo zagotovili, da so bile izhodne listine stopenj analiziranja in načrtovanja teh projektov primerljive po sestavi ter po nivoju podrobnosti podatkov, ki so bili navedeni v listinah.

Pri zbiranju podatkov o porabljenem času smo veljavnost pridobljenih podatkov zagotovili tako, da :

- smo opredelitev dejavnikov projektov (glejte preglednico 7) za vseh pet projektov opredelili sami, nato pa je sestavo preveril in potrdil še en član projektne skupine;
- smo sestave dela projekta (WBS) za vseh pet projektov opredelili sami, nato pa je sestavo preveril in potrdil še en član projektne skupine;
- smo na podlagi WBS projektov dvakrat prešteli število transkacij za posamezen projekt, nato pa je transakcije preštel in število potrdil še en član projektne skupine. Pri štetju transakcij smo uporabila opredelitev iz poglavja 4.1.4;
- smo podatke o porabljenem času za stopnji analiziranja in načrtovanja zbirali sami in to le za svoje delovne naloge. V primeru, da smo določen sklop (ali transakcijo) programske opreme analizirali ali načrtovali skupaj z drugim članom, smo ta sklop (ali transakcijo) izločili iz raziskave.

4.2 Opis izvedbe raziskave in zbiranja podatkov

Raziskava je bila izvedena med leti 2006 in 2008 v okviru moje redne zaposlitve pri takratnem delodajalcu. V teh treh letih sem sodeloval na več različnih projektih, kjer sem bil zadolžen predvsem za opravljanje delovnih nalog iz okvira stopenj analiziranja, načrtovanja ter preizkušanja programske opreme. Del teh projektov predstavlja tudi pet projektov, ki so vključeni v to raziskavo. Imena vseh petih projektov predstavljajo poslovno skrivnost podjetja, zato sem jih poimenoval z navideznimi imeni: projekt A1, projekt A2, projekt B1, projekt C1 in projekt C2 (glejte preglednico 7). Pri vseh petih projektih sem opravljal delovne naloge iz stopnje načrtovanja, pri štirih projektih sem opravljal naloge iz stopnje analiziranja, pri treh pa naloge iz stopnje preizkušanja programske opreme. Zaradi premajhne količine zbranih podatkov sem podatke o porabljenem času iz stopnje preizkušanja izločil iz raziskave. Izbrane projekte sem izvajal pod vplivom različnih dejavnikov (glejte preglednico 7), vendar so se pri nekaterih projektih pojavljali enaki dejavniki. Tako sem lahko projekte razdelil v dva razreda (glejte preglednico 9): spletna programska oprema (projekta A1 in A2) in registri (projekti B1, C1 in C2). To razdelitev sem uvedel z namenom, da preverim, ali lahko dosežem natančnejšo oceno potrebnega časa za razvoj programske opreme, če za določen projekt uporabim zgodovinske podatke porabljenega časa projektov iz istega razreda. Več o tem v nadaljevanju.

Kot že napisano, sem ob opravljanju svojih delovnih nalog sem sproti beležil porabljen čas. Vsebino delovne naloge je ponavadi predstavljal določen primer uporabe (PU), v nekaterih primerih pa je vsebino delovne naloge predstavljala listina, ki opisuje splošno, logično zaokroženo delovanje programske opreme, skupno večjemu številu PU. Primeri takšnih listin so: opis skupnega modela zbirke podatkov, opis skupnih organizacijskih predpisov ali funkcionalnih pravil itd. Po končanem zbiranju podatkov sem porabljen čas seštel po posameznih delovnih nalogah. Porabljen čas za listine, ki so bile skupne več PU, sem enakomerno porazdelil med porabljeni čas za posamezne PU. Naprimer: imel sem listino, ki je opisovala model zbirke podatkov in je bila skupna 10 PU. Za pripravo (načrtovanje) te listine sem porabil 100 enot časa. Ta čas sem porazdelil med 10 PU tako, da sem k porabljenemu času za načrtovanje vsakega PU dodelil 10 enot časa (desetino porabljenega časa za pripravo skupne listine). Na koncu sem seštel porabljen čas vseh PU po posamezni stopnji za vsak projekt. Tako sem dobil porabljen čas za posamezno stopnjo na projektu (glejte preglednico 9).

Projekt	Stopnja razvoja	Dejansko porabljen čas (v urah)	Število sklopov	Število PU	Število transakcij	Skupina
Projekt A1	Analiziranje	15,00	1	1	2	Spletna programska oprema
	Načrtovanje	17,00	1	1	2	
Projekt A2	Analiziranje	100,00	1	4	22	Spletna programska oprema
	Načrtovanje	181,50	1	4	22	
Projekt B1	Analiziranje	275,50	3	7	46	Register
	Načrtovanje	492,00	3	7	46	
Projekt C1	Analiziranje	108,50	6	7	15	Register
	Načrtovanje	804,00	10	32	89	
Projekt C2	Načrtovanje	82,50	4	5	11	Register

Preglednica 9: Podatki o porabljenem času in velikosti programske opreme iz projektov

Preglednica 9 prikazuje podatke iz sestave WBS za vsak projekt in stopnjo razvoja posebej. Podatki iz WBS se nahajajo v stolpcih »število sklopov«, »število PU« in »število transakcij« ter prikazujejo obseg delovanja programske opreme. Transakcije sem preštel na podlagi opredelitve iz poglavja 4.1.4. Če se osredotočimo zgolj na stolpec »število transakcij«, lahko ugotovimo, da ima najmanjši obseg programska oprema iz projekta A1, in sicer le 2 transakciji, največji obseg pa ima programska oprema iz projekta C1, in to kar 89 transakcij. Opazimo lahko tudi razliko med obsegom delovanja programske opreme pri stopnji analiziranja in stopnji načrtovanja projekta C1. Razlika je posledica dejstva, da pri stopnji analiziranja nisem sodeloval pri izdelavi vseh PU, zato sem v raziskavi upošteval le tiste transakcije, ki so bile zaznave v okviru PU, katere sem izdelal izključno sam. V okviru stopnje načrtovanje pa sem izdelal načrte za vse PU, zato sem v raziskavi za to stopnjo uporabil večje število transakcij kot za stopnjo analiziranja. Tudi stopnjo načrtovanja na projektu C2 nisem opravil v celoti sam. Med uresničevanjem projekta sem namreč uvajal novega zaposlenca – načrtovalca. Del svojih delovnih nalog sem dodelil novemu načrtovalcu, za opravljanje teh delovnih nalog pa zato nisem mogel zbrati podatkov o porabljenem času. Zaradi tega v raziskavo nisem vključil vseh transakcij, temveč le tiste, katere sem načrtoval izključno sam. Prav zato je obseg delovanja programske opreme projekta C2 ustrezno manjši.

Dejansko porabljen čas po projektih in posameznih stopnjah je v preglednici 9 predstavljen v urah. Vsebuje dejansko porabljen čas brez časa, porabljenega za počitek ali malico. Hiter pogled na preglednico 9 nam pove, da je obseg porabljenega časa za načrtovanje sorazmeren z obsegom delovanja projekta. Tako lahko opazimo, da sem za opravljanje nalog na najmanjšem projektu A1 porabil najmanj časa, in sicer 15 ur za stopnjo analiziranja in 17 ur za stopnjo načrtovanja, medtem ko sem za največja projekta B1 in C1 porabil največ časa. Za stopnjo načrtovanja projekta B1 sem porabil 492 ur, za stopnjo načrtovanja projekta C1 pa celo 804 ure oziroma dobrih 100 človek/dni.

4.3 Obdelava zbranih podatkov raziskave

4.3.1 Priprava ocene potrebnega časa na podlagi podatkov vseh uresničenih projektov

Osnovo za izvedbo te raziskave predstavlja delo avtorjev Robiole in Orosca (2008). Avtorja sta v okviru svoje raziskave pridobila ocene potrebnega časa na podlagi podatkov o porabljenem času vseh projektov, vključenih v njuno raziskavo. Če povemo drugače: šele ko so bili uresničeni vsi projekti in zbrani vsi podatki o porabljenem času, sta avtorja na podlagi zbranih podatkov izračunala ravni povprečne produktivnosti na projektih njune raziskave in na podlagi podatkov o povprečnih produktivnosti pripravila ocene potrebnega časa. Zaradi primerljivosti izidov raziskave omenjenih avtorjev in raziskave tega dela sem se odločil, da bom na enak način tudi sam pripravil ocene potrebnega časa. S tako obdelanimi podatki bom lahko potrdil (ali zavrnil) ustreznost metode in s tem dosegel tretji cilj tega dela.

V preglednici 10 so prikazani obdelani podatki za stopnjo analiziranja po posameznih projektih, v preglednici 11 pa so prikazani obdelani podatki za stopnjo načrtovanja po posameznih projektih. Stolpec »Produktivnost« prikazuje izračunano produktivnost na projektu glede na enačbo 25, pri čemer spremenljivka »velikost programske opreme« predstavlja število transakcij, spremenljivka »vložen trud« pa dejansko porabljen čas. Podatki v stolpcih »produktivnost na projektu« in »povprečna produktivnost« so izraženi z enoto »število transakcij na uro«. Stolpec MRE (*angl. magnitude of relative error*) predstavlja podatek o stopnji relativne napake ocene potrebnega časa.

Podatek povprečne produktivnosti nekega projekta sem izračunal kot povprečje produktivnosti ostalih projektov, neodvisno od zaporedja časovne izvedbe projektov. Naprimer: raven pričakovane povprečne produktivnosti stopnje analiziranja projekta A1 (0,175 transakcij/uro) sem izračunal tako, da sem seštel ravni produktivnosti stopnje analiziranja projektov A2 (0,220 transakcij/uro), B1 (0,167 transakcij/uro) in C1 (0,138 transakcij/uro), nato pa sem pridobljeno vsoto (0,525 transakcij/uro) delil s tri. Predzadnji stolpec predstavlja oceno potrebnega časa za posamezen projekt. Oceno, izraženo v urah, sem izračunal tako, da sem število transakcij delil s podatkom povprečne produktivnosti projekta.

Na koncu sem izračunal še stopnjo relativne napake (v nadaljevanju MRE) glede na enačbo 26. V enačbi spremenljivko »predviden vložek« predstavlja ocena potrebnega časa, spremenljivko »dejanski vložek« pa dejansko porabljeni čas. Če stopnjo relativne napake pomnožimo s 100, dobimo odstotek, ki nam pove, za koliko smo se zmotili pri pripravi ocene potrebnega časa. Zelena barva v preglednicah 10 in 11 prikazuje napako, ki je manjša ali enaka meji sprejemljivosti oz. 25 odstotkom, rdeča barva (v preglednici 10) pa prikazuje napako, večjo od meje sprejemljivosti.

Projekt	Dejansko porabljen čas (v urah)	Število transakcij	Produktivnost na projektu	Povprečna produktivnost	Ocena potrebnega časa	MRE
Projekt A1	15,00	2	0,133	0,175	11,43	0,238
Projekt A2	100,00	22	0,220	0,146	150,68	0,507
Projekt B1	275,50	46	0,167	0,164	280,49	0,018
Projekt C1	108,50	15	0,138	0,173	86,71	0,201

Preglednica 10: Prikaz ocene potrebnega časa in napake za stopnjo analiziranja na podlagi podatkov vseh uresničenih projektov

Projekt	Dejansko porabljen čas (v urah)	Število transakcij	Produktivnost na projektu	Povprečna produktivnost	Ocena potrebnega časa	MRE
Projekt A1	17,00	2	0,118	0,115	17,39	0,023
Projekt A2	181,50	22	0,121	0,114	192,98	0,063
Projekt B1	492,00	46	0,093	0,121	380,17	0,227
Projekt C1	804,00	89	0,111	0,116	767,24	0,046
Projekt C2	82,50	11	0,133	0,111	99,10	0,201

Preglednica 11: Prikaz ocene potrebnega časa in napake za stopnjo načrtovanja na podlagi podatkov vseh uresničenih projektov

V preglednici 12 sta prikazana podatka o kakovosti pripravljene ocene za posamezno stopnjo razvoja programske opreme. Kakovost pripravljene ocene je izračunana glede na enačbo 27. Za stopnjo analiziranja podatek pove, da je bila pri 75 odstotkih projektov napaka ocene potrebnega časa manjša ali enaka od 25 odstotkov, za stopnjo načrtovanja pa je bila pri vseh projektih napaka manjša od 25 odstotkov.

Stopnja	Kakovost priprave ocene pri meji sprejemljivosti 25 odstotkov
Analiziranje	0,75
Načrtovanje	1,00

Preglednica 12: Kakovost pripravljene ocene na podlagi podatkov vseh uresničenih projektov

4.3.2 Priprava ocene potrebnega časa na podlagi podatkov uresničenih podobnih projektov

V okviru raziskave nas je zanimalo, ali lahko povečamo kakovost ocene potrebnega časa tako, da projekte glede na njihovo podobnost združimo v razrede. Osnovna zamisel je ta, da se podobni projekti razvoja programske opreme uresničujejo ob podobnih dejavnikih (*angl. contextual factors*), zaradi česar lahko sklepamo, da je pri teh projektih za uresničitev stopenj analiziranja in načrtovanja za osnovne enote programske opreme oz. transakcije potreben podoben obseg časa. Seznam podobnih dejavnikov je naveden v preglednici 6. V danem primeru smo projekte, vključene v raziskavo, zato lahko razdelili v dva razreda: spletna programska oprema in registri (glejte preglednico 7). Projekta A1 in A2 uvrščamo med spletno programsko opremo, projekti B1, C1 in C2 pa med registre. Glede na to umestitev smo tudi ustrezno obdelali podatke. V preglednicah 13 in 14 smo podatke o produktivnosti,

oceni potrebnega časa in stopnji relativne napake (MRE) izračunali na enak način kot v prejšnjem poglavju, razlika se pojavlja le pri izračunu povprečne produktivnosti.

Za določen projekt smo v tem primeru povprečno produktivnost izračunali le na podlagi podatkov iz že uresničenih podobnih projektov. Tako (glejte preglednico 14) smo za projekt A1 izračunali povprečno produktivnost na podlagi povprečne produktivnosti projekta A2, ki je edini njemu podoben projekt – spletna programska oprema. V primeru projekta B1 pa smo povprečno produktivnost izračunali kot povprečje produktivnosti projektov C1 in C2, ki sta podobna projekta kot B1. Umeščamo ju namreč med registre.

Projekt	Dejansko porabljen čas (v urah)	Število transakcij	Produktivnost na projektu	Povprečna produktivnost	Ocena potrebnega časa	MRE
Projekt A1 (Spletna)	15,00	2	0,133	0,220	9,09	0,394
Projekt A2 (Spletna)	100,00	22	0,220	0,133	165,41	0,654
Projekt B1 (Register)	275,50	46	0,167	0,138	333,33	0,210
Projekt C1 (Register)	108,50	15	0,138	0,167	89,82	0,172

Preglednica 13: Prikaz ocene potrebnega časa in napake za stopnjo analiziranja na podlagi podatkov že uresničenih podobnih projektov

Projekt	Dejansko porabljen čas (v urah)	Število transakcij	Produktivnost na projektu	Povprečna produktivnost	Ocena potrebnega časa	MRE
Projekt A1 (Spletna)	17,00	2	0,118	0,121	16,53	0,028
Projekt A2 (Spletna)	181,50	22	0,121	0,118	186,44	0,027
Projekt B1 (Register)	492,00	46	0,093	0,122	377,05	0,234
Projekt C1 (Register)	804,00	89	0,111	0,113	787,61	0,020
Projekt C2 (Register)	82,50	11	0,133	0,102	107,84	0,307

Preglednica 14: Prikaz ocene potrebnega časa in napake za stopnjo načrtovanja na podlagi podatkov že uresničenih podobnih projektov

V preglednici 15 sta prikazana podatka o kakovosti pripravljene ocene za posamezno stopnjo razvoja programske opreme glede na podatke iz podobnih projektov. Izračun kakovosti je enak izračunu iz prejšnjega poglavja.

Stopnja	Kakovost priprave ocene pri meji sprejemljivosti 25 odstotkov
Analiziranje	0,50
Načrtovanje	0,80

Preglednica 15: Kakovost priprave ocene na podlagi podatkov že uresničenih podobnih projektov

4.3.3 Priprava ocene potrebnega časa na podlagi podatkov sproti uresničenih projektov

Na koncu smo želeli preveriti, kako na kakovost pripravljene ocene vpliva postopno, časovno urejeno zbiranje in uporaba podatkov iz že uresničenih projektov. Za razliko od obeh prejšnjih obdelav, pri katerih sem povprečno produktivnost za vse projekte izračunal na koncu, po uspešni uresničitvi vseh projektov, sem zdaj za vsak projekt izračunal povprečno produktivnost sproti, glede na takrat dostopne podatke o predhodno uresničenih projektih. V tem primeru za prvi projekt nisem mogel izračunati povprečne produktivnosti, saj podatki o predhodnem projektu niso obstajali. To je lepo razvidno iz preglednic 16 in 17, kjer pri projektu A1, ki smo ga uresničili najprej, nismo imeli na voljo podatkov o povprečni produktivnosti (na neobstoječem predhodnem projektu) in posledično o oceni potrebnega časa ter MRE. Za izračun povprečne produktivnosti projekta A2 sem uporabil podatke projekta A1, za npr. projekt C1 pa podatke projektov A1, A2 in B1. Pri tem načinu je seveda pomembno časovno zaporedje uresničevanja projektov. Projektov v tem primeru nisem razdelil v skupine.

Omenjeni način zbiranja podatkov predstavlja začetno točko, iz katere lahko izhaja združba, ko želi začeti pripravljati in uporabljati ocene potrebnega časa za posamezne stopnje razvoja programske opreme na projektih ali celo za celoten proces razvoja programske opreme. Kot vidimo v preglednicah 16 in 17, lahko dobimo dobro oceno za obe stopnji razvoja programske opreme že na podlagi dveh predhodno uresničenih projektih.

Projekt	Dejansko porabljen čas (v urah)	Število transakcij	Produktivnost na projektu	Povprečna produktivnost	Ocena potrebnega časa	MRE
Projekt A1	15,00	2	0,133	*Ni na voljo*	*Ni na voljo*	*Ni na voljo*
Projekt A2	100,00	22	0,220	0,133	165,41	0,654
Projekt B1	275,50	46	0,167	0,177	259,89	0,057
Projekt C1	108,50	15	0,138	0,173	86,71	0,201

Preglednica 16: Prikaz ocene potrebnega časa in napake za stopnjo analiziranja na podlagi podatkov sproti uresničenih projektov

Projekt	Dejansko porabljen čas (v urah)	Število transakcij	Produktivnost na projektu	Povprečna produktivnost	Ocena potrebnega časa	MRE
Projekt A1	17,00	2	0,118	*Ni na voljo*	*Ni na voljo*	*Ni na voljo*
Projekt A2	181,50	22	0,121	0,118	186,44	0,027
Projekt B1	492,00	46	0,093	0,120	383,33	0,221
Projekt C1	804,00	89	0,111	0,111	801,80	0,003
Projekt C2	82,50	11	0,133	0,111	99,10	0,201

Preglednica 17: Prikaz ocene potrebnega časa in napake za stopnjo načrtovanja na podlagi podatkov sproti uresničenih projektov

V preglednici 18 sta prikazana podatka o kakovosti pripravljene ocene za posamezno stopnjo razvoja programske opreme glede na podatke iz sproti uresničenih projektov. Izračun kakovosti je enak izračunu iz prejšnjega poglavja.

Stopnja	Kakovost priprave ocene pri meji sprejemljivosti 25 odstotkov
Analiziranje	0,67
Načrtovanje	1,00

Preglednica 18: Kakovost priprave ocene na podlagi podatkov sproti uresničenih projektov

4.3.4 Razčlenitev izidov raziskave in doseženih ciljev

4.3.4.1 Razčlenitev izidov stopnje analiziranja razvoja programske opreme

Če najprej razčlenimo podatke za pripravo ocene potrebnega časa stopnje analiziranja, pridobljene na podlagi vseh uresničenih projektov, takoj opazimo, da pri projektu A2 napaka ocene presega stopnjo sprejemljivosti 25 odstotkov. Najmanjšo napako smo dobili pri projektu B1, medtem ko pri projektih A1 in C1 napaka znaša dobrih 20 odstotkov. Nekoliko drugačno sliko dobimo pri razčlenitvi podatkov, zbranih na podlagi ocen potrebnega časa podobnih projektov. V tem primeru dobimo za vse projekte, povezane z razvojem spletne programske opreme (projekta A1 in A2), napako, ki je večja od meje sprejemljivosti. Pri projektu A2 je napaka znašala celo 65 odstotkov. Na drugi strani pa je napaka pri vseh projektih za razvoj registrov (projekta B1 in C1) znotraj meje sprejemljivosti. Pri pripravi ocen časa na podlagi sproti uresničenih projektov smo dobili podobne izide kot pri pripravi ocen na podlagi vseh uresničenih projektov. Pri projektu A2 napaka presega mejo sprejemljivosti in to kar za 65 odstotkov, najmanjšo napako pa smo dobili pri projektu B1, in sicer napaka znaša dobrih pet odstotkov, kar znaša približno 15 ur.

Iz pridobljenih podatkov lahko sklepamo, da z izbrano metodo za stopnjo analiziranja programske opreme dobimo oceno (potrebnega časa) ustrezne kakovosti le v primeru, ko pripravljamo oceno na podlagi podatkov vseh uresničenih projektov. V tem primeru smo z metodo lahko pripravili oceno za 75 odstotkov projektov, pri katerih je napaka manjša od 25 odstotkov. Za pripravo ocene na podlagi podatkov podobnih projektov smo z metodo lahko pripravili dovolj kakovostno oceno zgolj za polovico projektov, pri ocenjevanju na podlagi sproti uresničenih projektov pa smo lahko pripravili oceno ustrezne kakovosti za dve tretjini projektov. Ugotovimo lahko, da v tem primeru podobnost, glede na število projektov, sama ne zagotavlja nujno boljših izidov. V preglednici 13 se največja napaka pojavi pri projektu A2, pri katerem smo za pripravo ocene uporabili podatke podobnega projekta A1. Za razliko pa imamo v preglednici 16 najboljšo oceno pri projektu B1, pri katerem smo za pripravo ocene uporabili podatke zgolj iz projektov A1 in A2, ki jih, glede na podobnost, ne uvrščamo v isti razred kot projekt B1.

Dobljeni podatki nakazujejo, da dobimo boljše izide, če za pripravo ocen potrebnega časa stopnje analize uporabimo pristopa ocenjevanja na podlagi vseh uresničenih projektov in na podlagi sproti uresničenih projektov. Pri obeh pristopih smo pri ocenjevanju uporabili večje število predhodno uresničenih projektov kot pri ocenjevanju časa na podlagi uresničenih podobnih projektov. Slednje izide si lahko razložimo s krivuljo učenja, kot jo navaja Konečnik ([21] 2001; 390). Razlaga krivulje učenja temelji na človekovi sposobnosti (oz. sposobnosti združbe) učenja. S ponavljanjem ljudje v združbi opravijo delo hitreje in bolj učinkovito ter bolj kakovostno. V našem primeru število ponovitev predstavlja število uporabljenih uresničenih projektov, hitrost izvedbe dela pa kakovost ocene potrebnega časa. V našem primeru se s številom uresničenih projektov povečuje kakovost ocene potrebnega

časa. Iz podatkov (glejte preglednico 16) lahko tudi sklepamo, da je za pripravo boljših ocen časa bolj pomemben dejavnik »število že uresničenih projektov« kot pa dejavnik »podobnost med projekti«, ki smo ga opredelili v tem delu. V prihodnje bi bilo potrebno raziskati vpliv dejavnika »podobnosti med projekti« na pripravo ocene časa v stopnji analiziranja na večjem vzorcu projektov. Možno je, da bodo prihodnje raziskave glede na težnjo izboljševanja napovedovanja pri sproti uresničenih projektih v preglednici 16 (napaka je večja od 25 odstotkov) pri večjem številu projektov nakazale izide, ki bi potrdili ustreznost izbrane metode za pripravo ocene v stopnji analiziranja za pristop ocenjevanja na podlagi podobnih projektov. Prav tako bi bilo v prihodnje potrebno raziskati še druge dejavnike, ki lahko opredelijo podobnost med projekti in bi lahko vplivali na pripravo ocene potrebnega časa pri stopnji analiziranja programske opreme.

Če povzamemo: ustreznost uporabljene metode za stopnjo analiziranja je dokazana zgolj pri pripravi ocene na podlagi že uresničenih projektov (glejte preglednico 12), pri pripravi ocene na podlagi podobnih projektov in pri sprotnem uresničevanju projektov pa metoda ni ustrezna, saj je v teh dveh primerih njena kakovost priprave ocene manjša od 0,75 oz. meje sprejemljivosti napake. Slednje si lahko razlagamo s tem, da smo pri teh dveh primerih uporabili premajhen vzorec projektov za pripravo ocene.

4.3.4.2 Razčlenitev izidov stopnje načrtovanja razvoja programske opreme

Pri ocenjevanju potrebnega časa za stopnjo načrtovanja sem dobil veliko boljše izide. V primeru ocenjevanja na podlagi vseh uresničenih projektov sem dobil pri vseh projektih oceno potrebnega časa, ki se nahaja znotraj meje sprejemljivost 25 odstotkov. Najmanjšo napako sem dobil pri projektu A1 (2,3 odstotka), največjo pa pri projektu B1 (22,7 odstotka). Kakovost pripravljene ocene za ta pristop ocenjevanja dosega najvišjo možno vrednost 1 (glejte preglednico 12), kar potrjuje ustreznost izbrane metode.

Tudi pri ocenjevanju na podlagi podatkov uresničenih podobnih projektov smo dobili na štirih projektih ocene, katerih napaka ocene je manjša od 25 odstotkov. Ti projekti so: A1, A2, B1 in C1 (glejte preglednico 14). Nekoliko odstopa ocena časa projekta C2, katerega napaka ocene znaša dobrih 30 odstotkov. Slednje si lahko razložimo na enak način kot pri stopnji analiziranja – s premajhnim številom uresničenih projektov, ki smo jih uporabili pri pripravi ocene potrebnega časa. Sicer pa je raziskava potrdila ustreznost ocenjevanja časa na podlagi uresničenih podobnih projektov, saj je kakovost pripravljene ocene večja od 0,75 (glejte preglednico 15).

Dobre izide oz. ocene potrebnega časa sem dobil pri ocenjevanju na podlagi podatkov sproti uresničenih projektov. Pri vseh projektih je napaka ocene manjša od meje sprejemljivosti, s čimer smo z raziskavo potrdili ustreznost uporabljenega pristopa pri pripravi ocene potrebnega časa. Kakovost pripravljene ocene za ta pristop ocenjevanja znaša 1 (glejte preglednico 18).

Če povzamemo dobljene ugotovitve, lahko zapišemo, da smo z raziskavo nakazali ustreznost uporabljene metode za stopnjo načrtovanja pri vseh treh pristopih ocenjevanja potrebnega časa. Za razliko od stopnje analiziranja smo v tem primeru dobili ustrezno kakovost pripravljene ocene tudi za ocenjevanje na podlagi podatkov iz podobnih projektov. Iz slednjega lahko sklepamo, da je možno boljše ocenjevanje potrebnega časa z združevanjem podobnih projektov v razrede. Slednje je potrebno dodatno raziskati in potrditi na večjem

številu projektov. Prav tako je kljub dobrim izidom raziskave potrebno dodatno raziskati vpliv obstoječih ter novih dejavnikov, s katerimi opredelimo podobnost projektov, na pripravo ocene potrebnega časa za stopnjo načrtovanja programske opreme.

5 Sklepne ugotovitve

V okviru te naloge in raziskave sem si zadal tri cilje. Prvi cilj je bil preučiti metode za ocenjevanje potrebnega časa, še posebej metodo poenostavljenih točk primerov uporabe. V okviru drugega cilja sem si zadal nalogo, da izberem pet projektov, na katerih bom izvedel raziskavo. Tretji cilj je zajel izvedbo raziskave, v okviru katere naj bi potrdil ustreznost kakovosti oz. natančnosti pripravljene ocene potrebnega časa na podlagi metode poenostavljenih točk primerov uporabe in to le za stopnji analiziranja in načrtovanja programske opreme. Sprejemljivost ocene je določena s tem, da napaka ocene ne sme presegati 25 odstotkov dejansko porabljenega časa pri vsaj 75 odstotkih projektov.

Prvi cilj sem dosegel v okviru 2. in 3. poglavja, kjer sem podrobneje predstavil dosedanja spoznanja s področja ocenjevanja časa. V okviru teh poglavij sem predstavil sestavo WBS oz. način opisa in predstavitve obsega delovanja programske opreme. Ta sestava je obvezno orodje, ki ga potrebujemo, če želimo pripraviti oceno porabljenega časa na sistematičen in kakovosten način. Drugi cilj sem dosegel tako, da sem izbral projekte, na katerih sem med letoma 2006 in 2008 sodeloval v okviru svoje zaposlitve pri takratnem delodajalcu. Na projektih A1, A2, B1, C1 in C2 sem lahko nepristransko in natančno zbiral podatke o porabljenem času za stopnjo analiziranja in načrtovanja programske opreme, s čimer sem pridobil ustrezno izhodišče za izvedbo raziskave.

Tretji cilj sem dosegel na podlagi ugotovitev, pridobljenih iz obdelave in analize zbranih podatkov raziskave. V okviru tega cilja sem si zastavil en glavni in dva stranska podcilja. Glavni podcilj je bil potrditi ugotovitve avtorjev Robiole in Orosca ([35] 2008) o ustreznosti uporabe izbrane metode za ocenjevanje potrebnega časa, vendar ne za celoten razvoj programske opreme, temveč zgolj za izbrane stopnje razvoja programske opreme, kot sta analiziranje in načrtovanje. Drugi podcilj je bil potrditi tezo, da izbrana metoda omogoča pripravo ustrezne ocene potrebnega časa na podlagi podobnih uresničenih projektov in to za stopnjo analiziranja ter načrtovanja. Podobni projekti so bili tisti, ki sem jih lahko na podlagi izbranih dejavnikov razvrstil v isti razred. Razred »spletna programska oprema« sta predstavljala projekta A1 in A2, razred »registri« pa projekti B1, C1 in C2. V okviru tretjega podcilja pa sem želel potrditi možnost uporabe metode pri pripravi ocen potrebnega časa za stopnji analiziranja in načrtovanja na podlagi sproti uresničenih projektov.

Na podlagi ugotovitev raziskave lahko zapišemo, da je bil tretji cilj dosežen, saj smo z metodo pripravili ocene potrebnega časa za stopnjo analiziranja in načrtovanja v okviru želene kakovosti (glejte preglednico 12). S tem smo tudi potrdili predhodne ugotovitve Orosca in Robiole ([35] 2008) o ustreznosti uporabe metode poenostavljenih točk primerov uporabe za pripravo ocen potrebnega časa. Drugi in tretji podcilj smo na podlagi zbranih podatkov in ugotovitev le delno potrdili, saj je ustreznost metode dokazana za stopnjo načrtovanja, ne pa tudi za stopnjo analiziranja (glejte preglednice 15 in 18). Glede na spodbudne izide iz ocenjevanja na podlagi vseh uresničenih projektov pa lahko sklepamo, da lahko z večjim naborom projektov dobimo manjše razlike med načrtovanim oz. predvidenim in dejansko porabljenih časom tudi za ostala dva pristopa. Pri ocenjevanju časa na podlagi vseh uresničenih projektov sem lahko uporabil podatke že vseh petih projektov, kar je verjetno vplivalo tudi na natančnejše ocene. Pri ocenjevanju časa na podlagi podobnih projektov sem za stopnjo analiziranja uporabil podatke le o dveh, pri stopnji načrtovanja pa le o treh projektih. Uporaba premajhnega nabora projektov je razvidna tudi pri ocenah, saj je napak ocen veliko, hkrati pa so napake le-teh dokaj visoke (glejte preglednice 13 in 14). Pri

ocenjevanju časa na podlagi sproti uresničenih projektov sem prav tako uporabil majhen nabor projektov, saj sem za prvo pripravo ocene časa (projekt A2) uporabil podatke enega projekta, pri drugi pripravi ocene časa (projekt B1) sem uporabil podatke iz dveh projektov itd. (glejte preglednice 16 in 17). Iz podatkov je razvidno, da se prevelika napaka ocene časa v stopnji analiziranja pojavi zgolj pri prvi pripravi ocene (projekt A2). Pri ostalih ocenah se napaka nahaja znotraj meje sprejemljivosti. Iz tega lahko sklepamo, da je zgolj en projekt predstavljal premajhno osnovo za pripravo ocene časa projekta A2, hkrati pa lahko sklepamo, da bi lahko uporaba večjega nabora projektov dokazala ustreznost uporabe metode tudi za stopnjo analiziranja programske opreme. Podatki raziskave tako nakazujejo, da ima na kakovost ocene porabljenega časa večji vpliv dejavnik »število« kot pa »podobnost« med projekti. Ta ugotovitev predstavlja spodbudo, da se lahko z ocenjevanjem časa z metodo poenostavljenih točk primerov uporabe nemudoma »spoprimejo« tudi podjetja brez lastnih izkušenj s področja ocenjavanja potrebnega časa. Metoda namreč na enostaven in hiter način ponuja sprejemljive izide že pri majhnem naboru uresničenih ter dokaj raznovrstnih projektov.

Z ocenjevanjem časa na podlagi vseh uresničenih projektov sta tako raziskava Robiole in Orosca ([35] 2008) in tu opravljena raziskava dokazala pozitivno povezavo med količino zbranih podatkov o porabljenem času iz preteklih projektov ter natančnostjo oz. kakovostjo pripravljene ocene potrebnega časa ([10] Heldman in drugi, 2005; 259). Ta ugotovitev velja tako za celoten razvoj programske opreme kot tudi za posamezne stopnje, kot sta analiziranje in načrtovanje programske opreme.

V prihodnje bi bilo potrebno raziskavo iz tega dela ponoviti na večjem, statistično sprejemljivem, naboru projektov. Avtorji te raziskave bi lahko ponudili statistično značilne izide, ki bi dokazali sprejemljivost opisane metode, in tako upravičili njeno uporabo v praksi. Prav tako bi bilo potrebno na večjem naboru projektov dodatno raziskati vpliv dejavnikov, ki opredeljujejo podobnost med projekti. Priporočljivo bi bilo, da bi bili v raziskavo vključeni dejavniki, ki so navedeni v tem delu (navedeni v preglednici 6). Namen te raziskave bi bil preučiti vpliv omenjenih dejavnikov na kakovost ocene potrebnega časa. V prihodnje bi lahko raziskovalci preučili uporabo metode poenostavljenih točk primerov uporabe tudi za ostale stopnje razvoja programske opreme, kot so zajem zahtev, izvedba in preizkušanje. V okviru take raziskave bi lahko preučili tudi vpliv dejavnikov, ki opredeljujejo podobnosti projektov, na pripravo ocene potrebnega časa.

6 Literatura in viri

6.1 Literatura

[1] ANDA, Bente; BENESTAD C. Hans; HOVE E. Siw: A Multiple – Case Study of Software Effort Estimation based on Use Case Points, 2005 International Symposium on Empirical Software Engineering, Noosa Heads, Australia, 2005, str. 407–416.

[2] BAJEC, Marko; VAVPOTIČ, Damjan; KRISPER, Marjan: Practice-driven approach for creating project-specific software development methods, Information and Software Technology, št. 4, zv. 49, april, 2007, str. 345–365.

[3] BITTNER, Kurt; SPENCE Ian: Managing Iterative Software Development Projects, Addison-Wesley, Boston, 630 strani.

[4] BRAZ, Rodrigo Marcio; VERGILIO, Regina Silvia: Software Effort Estimation Based on Use Cases, 30. Annual International Computer Software and Applications Conference (COMPSAC'06), zv. 1, Chicago, Illinois, USA, 2006, str. 221–228.

[5] BUNC, Stanko: Slovar tujk, Založba obzorja Maribor, Maribor, 1965, 472 strani.

[6] CACHIA, Ernest; MICALLEF Mark: Measuring the Functionality of Online Stores, Proceedings of the 29. Annual International Computer Software and Applications Conference, zv. 02, Edinburgh, Scotland, 2005, str. 70–73.

[7] DEVAUX, A. Stephen: Total Project Control, A Manager's Guide to Integrated Project Planning, Measuring and Tracking, John Wiley & Sons, Inc., New York, 1999.

[8] DREIEM, Hege; ANDA, Bente; SJØBERG, I. K. Dag; JØRGENSEN, Magne: Estimating Software Development Effort Based on Use Cases – Experiences from Industry, Zbornik konference »27. International Conference on Software engineering«, St. Louis, Missouri, USA, 2005, str. 303–311.

[9] FITZSIMMONS, James A.; FITZSIMMONS, Mona J.: Service Management, Operations, Strategy and Information Technology, Second Edition, McGraw-Hill, Boston, 1998, 613 strani.

[10] HELDMAN, Kim; BACA, Claudia; JANSEN, Patti: Project Management Professional, Study Guide, Deluxe Edition, Wiley Publishing, Inc., New Jersey, 2005, 768 strani.

[11] HENRY M. Raymond; McCRAY E. Gordon; PURVIS L. Russell; ROBERTS L. Tom: Exploiting organizational knowledge in developing IS project cost and schedule estimates: An empirical study, Information and Management, 44, 2007, str. 598–612.

[12] HERIČKO, Marjan; ŽIVKOVIČ Aleš: The size and effort estimates in iterative development, Information and Software Technology, zv. 50, št. 7–8, 2008, str. 772–781.

[13] HERMONE, H. Ronald: The Management Survival Manual for Engineers, CRC Press, Boca Raton, 1998.

[14] HOPE, Christine; MÜHLEMANN, Alan: Service Operations Management, Strategy, Design and Delivery, Prentice Hall, London, 1997.

[15] ISSA, Ayman; ODEH, Mohammed; COWARD, David: Software Cost Estimation Using Use – Case Models: a Critical Evaluation, Information and Communication Technologies, 2006. ICTTA '06., str. 2766–2771.

[16] JOHNSTON, Rober; GRAHAM, Clark: Service Operations Management, Prentice Hall, Harlow, 2001, 413 strani.

[17] JØRGENSEN, Magne; SHEPPERD, Martin: A Systematic Review of Software Development Cost Estimation Studies, IEEE Transactions on Software Engineering, zv. 33, št. 1, januar, 2007, str. 33–53.

[18] JØRGENSEN, Magne; MOLØKKEN – ØSTVOLD, Kjetil: Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and

Data Analysis Method, IEEE Transactions on Software Engineering, zv. 30, št. 12, december, 2004, str. 993–1007.
[19] KERZNER, Harold: Project Management: Case Studies, Second Edition, John Wiley & Sons, Inc., New Jersey, 2006, 659 strani.
[20] KOMAC, Daša: Angleško-slovenski, slovensko-angleški slovar, četrta izdaja, Cankarjeva založba, Ljubljana, 1996, 935 strani.
[21] KONEČNIK, Maja: Proces in krivulja učenja podjetja, Organizacija, letnik 34, številka 6, 2001 str. 389–396.
[22] KUSUMOTO, Shinji; MATUKAWA, Fumikaze; INOUE, Katsuro; HANABASU, Shigeo; MAEGAWA, Yusuke: Estimating Effort by Use Case Points: Method, Tool and Case Study, 10. IEEE International Symposium on Software Metrics (METRICS'04), Chicago, USA, 2004, str. 292–299.
[23] MANOGHEGHI, Parastoo; ANDA, Bente; CONRADI, Reidar: Effort Estimation of Use Cases for Incremental Large-Scale Software Development, 27. International Conference on Software engineering, St. Louis, Missouri, USA, 2005, str. 303–311.
[24] MATSON E, Jack; BARRETT E., Bruce; MALLICHAMP M., Joseph: Software Development Cost Estimation Using Function Points, IEEE Transactions on Software Engineering, zv. 22, št. 4, april, 1994, str. 275–287.
[25] McGARRY, John; CARD, David; JONES, Sheryl; LAYMAN, Beth; CLARK, Elizabeth; DEAN, Joseph; HALL, Fred: Practical Software Measurement: Objective Information for Decision Makers, Addison-Wesley, Boston, 2002, 277 strani.
[26] MELI, Roberto; SANTILO, Luca: Function point estimation methods: A comparative overview, The European Software Measurement Conference, FESMA 98, Antwerp, 1999.
[27] MENZIES, Tim; CHEN, Zhihao; HIHN, Jairus; LUM, Karen: Selecting Best Practices for Effort Estimation, IEEE Transactions on Software Engineering, zv. 32, št. 11, november, 2006, str. 883–895.
[28] MIHELČIČ, Miran: Vloga listin – organizacijskih predpisov pri zagotavljanju koordinacije, 11. znanstveno posvetovanje o organizaciji: Koordinacijski in komunikacijski vidiki organizacije združb, junij, 2010, str. 16-26.
[29] MIHELČIČ, Miran: Organizacija in ravnateljevanje, Fakulteta za računalništvo in informatiko, Ljubljana, 1999, 592 strani.
[30] MIHELČIČ, Miran: Organizacija in ravnateljevanje, Fakulteta za računalništvo in informatiko, Ljubljana, 2008, 554 strani.
[31] MIHELČIČ, Miran: Poslovne funkcije, Fakulteta za računalništvo in informatiko, Ljubljana, 2004, 363 strani.
[32] MOLØKKEN-ØSTVOLD, Kjetil; JØRGENSEN, Magne: A Comparison of Software Project Overruns – Flexible versus Sequential Development Models, IEEE Transactions on Software Engineering, zv. 31, št. 9, september, 2005, str. 754–766.
[33] PAYNE C. Andrew, CHELSOM V. John, REAVILL R. P. Lawrence: Management for Engineers, John Wiley & Sons, Inc., Chicester, 1996, 506 strani.
[34] POW-SANG, Jose Antonio; JOLAY-VASQUEZ, Enrique: An Approach of a Technique for Effort Estimation of Iterations in Software Projects, 13. Asia Pacific Software Engineering Conference (APSEC'06), Bangalore, India, 2006, str. 367–376.
[35] ROBIOLO, Gabriela; OROSCO Ricardo: Employing use cases to early estimate effort with simpler metrics, Innovations in Systems and Software Engineering, zv. 4, št. 1, april, 2008, str. 31–43.
[36] ROZMAN, Rudi; STARE, Aljaž: Projektni management ali ravnateljevanje projekta, Ekonomska fakulteta, Ljubljana, 2008, 299 strani.

[37] VINSEN, Kevin; JAMIESON, Diane; CALLENDER, Guy: Use Case Estimation – The Devil is in the Detail, 12. IEEE International Requirements Engineering Conference (RE'04), Kyoto, Japan, 2004, str. 10–15.
[38] VRHOVEC, Luka: Prilagoditev informacijske podpore združenima procesoma izvedbe naročila informacijske in telekomunikacijske storitve, magistrska naloga, Ljubljana: Fakulteta za računalništvo in informatiko, 2009, 116 strani.
[39] WRIGHT, J. Nevan; RACE, Peter: The Management of Service Operations, druga izdaja, Thomson, London, 2004.
[40] ŽVANUT, Boštjan: Konstrukcija prilagojenih IT procesov na osnovi socio-tehničnih značilnosti obravnavane združbe, doktorska disertacija, Ljubljana, 2009, 160 strani.

6.2 Ostali viri

[41] (2009) *Wikipedia: COCOMO*. Najdeno na spletu dne 10.7.2010. Dostopno na: http://en.wikipedia.org/wiki/Cocomo
[42] (2008) *iRise Survey*. Najdeno na spletu dne 10.7.2010. Dostopno na: http://www.reuters.com/article/pressRelease/idUS114597+24-Jun-2008+BW20080624
[43] (2009) *Slovar informatike.* Najdeno na spletu dne 10.7.2010. Dostopno na: http://www.islovar.org
[44] (2009) *Wikipedia: Software development effort estimation*. Najdeno na spletu dne 10.7.2010. Dostopno na: http://en.wikipedia.org/wiki/Software_development_effort_estimation
[45] (2009) *Standish group's CHAOS Reports*. Najdeno na spletu dne 10.7.2010. Dostopno na: http://www.irise.com/blog/index.php/2009/06/08/2009-standish-group-chaos-report-worst-project-failure-rate-in-a-decade/
[46] (2005) *LONGSTREET, David: Fundamentals of Function Point Analysis*. Najdeno na spletu dne 10.7.2010. Dostopno na: http://www.softwaremetrics.com/files/Fundamentals%20of%20Function%20Point%20Analysis.pdf
[47] (2010) *Wikipedia: Earned value management*. Najdeno na spletu dne 19.11.2010. Dostopno na: http://en.wikipedia.org/wiki/Earned_value_management
[48] (2010) *Wikipedia: History of software engineering*. Najdeno na spletu dne 21.11.2010. Dostopno na: http://en.wikipedia.org/wiki/History_of_software_engineering
[49] (2010) *Wikipedia, Software development process*, Najdeno na spletu dne 21.11.2010. Dostopno na: http://en.wikipedia.org/wiki/Software_development_process
[50] (2009) *Wikipedia: Function Point*. Najdeno na spletu dne 10.7.2010. Dostopno na: http://en.wikipedia.org/wiki/Function_point