UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

Jovan M. Popović

UNAPREĐENJE METODA ZA PROCENU NAPORA SOFTVERSKIH PROJEKATA

DOKTORSKA DISERTACIJA

Beograd, 2016

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRICAL ENGINEERING

Jovan M. Popović

ENHANCING METHODS FOR SOFTWARE EFFORT ESTIMATION

Doctoral Dissertation

Belgrade, 2016

PODACI O MENTORU I ČLANOVIMA KOMISIJE

Mentor:

Dr Dragan Bojić, vanredni profesor, Univerzitet u Beogradu – Elektrotehnički fakultet

Članovi komisije:

Dr Boško Nikolić, vanredni profesor, Univerzitet u Beogradu – Elektrotehnički fakultet

Dr Dušan Starčević, redovni profesor, Univerzitet u Beogradu – Fakultet organizacionih nauka

Datum odbrane: _____.

ZAHVALOST

Želeo bih da se zahvalim profesoru dr Draganu Bojiću zato što je prihvatio da bude mentor u ovoj relativno novoj oblasti softverskog inženjerstva. S obzirom da je teško pronaći eksperte u ovoj oblasti kako u softverskoj industriji tako i u akademskim krugovima, mentorski rad na ovakvom istraživanju je predstavljao veliki izazov.
Posebnu zahvalnost bih uputio mr Nenadu Koroliji koji je aktivno učestvovao u istraživanju i pripremi rada koji smo zajedno sa prof. Bojićem objavili.
Konačno, zahvalio bih se svojoj porodici na podršci i vremenu kojim su mi omogućili da završim ovo istraživanje.

UNAPREĐENJE METODA ZA PROCENU NAPORA SOFTVERSKIH PROJEKATA

Rezime

Procena napora potrebnog da se implementira softverski sistem je jedan od najtežih ali i najbitnijih zadataka koji se postavlja pred softverske timove. Procena napora je često ključni faktor koji će odrediti da li će projekat početi, pod kojim ograničenjima će se raditi i da li će se uopšte uspešno završiti i koristiti. Realnost je da se procene često vrše na osnovu nedovoljno definisanih zahteva pomoću različitih heuristika, pretpostavki i iskustava, pri čemu se za ograničeno vreme traže što je moguće bolje procene.
I pored značajnog napretka u različitim oblastima softverskog inženjerstva tokom poslednjih decenija, procene napora se uglavnom i dalje svode na subjektivne procene timova ili eksperata u nekoj oblasti sa greškama procene koje često nisu zadovoljavajuće. Globalne statistike govore da se samo trećina softverskih projekata uspešno završi, trećina se završi uz probijanje budžeta i rokova, dok trećina projekata propadne. Jedan od glavnih razloga za ovakve loše rezultate su nedovoljno dobre procene napora.
Cilj ovog rada je analiza postojećih metoda procene i predlaganje novih metoda kojima se procene napora mogu olakšati i poboljšati. Osnovna ideja koja će biti predstavljena u radu je mogućnost uvođenja novih predikcionih modela i tehnika kojima će se postojeće metode poboljšati. U radu će biti predstavljeni predlozi rešenja za unapređenje postojećih metoda procene kao i konkretni rezultati koji se mogu dobiti primenom tih rešenja na realnim softverskim projektima. Unapređenja se svode na identifikaciju parametara kojima se mogu opisati projekti koji najviše utiču na napor potreban da se implementira projekat, kao i zavisnosti među parametrima i naporom.

Ključne reči: Procene napora, softverski sistemi, merenje

Naučna oblast: Tehničke nauke - Elektrotehnika

Uža naučna oblast: Softversko inženjerstvo

ENHANCING METHODS FOR SOFTWARE EFFORT ESTIMATION

Abstract

Estimating the effort required to implement the software system is one of the most difficult, but also the most important tasks in software engineering. Effort estimate is often a key factor that will indicate whether the project will begin, under which restrictions will be implemented and would it be successfully completed and used. The reality is that the estimates are often made based on the incomplete requirements using various heuristics, assumptions and experience, where best possible estimates are required in a limited timeframe.

Besides significant advances in various domains of software engineering in recent decades, effort estimation methods are mostly based on the subjective assessment by the project teams or experts in a certain field, with the effort estimate errors often higher than the expected. Global statistics show that only one third of software projects is successfully concluded, one third is not finished within the budget and deadlines, while the rest of projects fail. One of the main causes for these results are inaccurate effort estimates.

The aim of this thesis is to analyze the existing effort estimation methods and propose new methods that can improve effort estimation. The basic idea that will be presented in the thesis is the usage of new models and techniques that will improve the existing effort estimation methods. In the thesis, proposals for the improvement of existing assessment methods and concrete results that can be obtained using these solutions to real software projects will be presented. Improvements are confined to the identification of project parameters describing projects that influence the effort the most, as well as determining dependencies between the parameters and the effort.

Keywords: Effort estimation, Software systems, Measurements

Scientific field: Technical science – Electrical engineering

Specific scientific field: Software engineering

SADRŽAJ

INDEKS SLIKA

INDEKS TABELA

SPISAK SKRAĆENICA

SKR	Skraćenica
BOEM-RS	Tačka specifikacije zahteva po Boemovom modelu
COBRA	Cost Estimation, Benchmarking, and Risk Assessment
COCOMO II	Constructive Cost Model II
COSMIC	Common Software Measurement International Consortium
FP	Broj funkcionalnih tačaka
FPA	Analiza funkcionalnim tačkama
FZ	Funkcionalni zahtevi
GSC	Globalne karakteristike sistema
IFPUG	International Function Point Users Group
LOC	Broj linija koda
MK2	Mark II metoda
MLP	Višeslojna neuralna mreža
MMRE	Srednja relativna greška
MRE	Relativna greška
NESMA	Netherlands Software Metrics Association
NESMA-E	NESMA procenjena metoda
NESMA-I	NESMA indikativna metoda
NFZ	Nefunkcionalni zahtevi
PMI	Project Management Institute
PMI-B	Budžetske procene po PMI kriterijumu
PMI-I	Inicijalne procene po PMI kriterijumu
RUP	Rational Unified Process
UCP	Broj tačaka slučajeva korišćenja
UUCP	Neprilagođena UCP veličina
UP	Unified Process
XP	Ekstremno programiranje

1. Uvod

U procesu razvoja softvera jedna od prvih i najbitnijih aktivnosti je procena vremena ili napora koji je potrebno uložiti u softverski projekat. Veoma često softverski projekti počinju sledećom rečenicom:

„Koliko bi bilo potrebno vremena da se…?“

U trenutku postavljanja pitanja, zahtevi nisu definisani, verovatno ne postoji ni jasna ideja šta bi trebalo da se napravi, otvorene su sve opcije vezane za tehnologiju i dizajn. Međutim i dalje je glavno pitanje: „Koliko je vremena potrebno?“.

Vreme izrade projekta ili delova projekata je izuzetno bitno klijentima i menadžerima zato što ono govori koliko će novca biti potrebno uložiti u projekat, a samim tim utiče i na odluku da li počinjati projekat ili ne. Ako je novac koji je potrebno uložiti približan ili čak i veći od očekivane dobiti, procene mogu odmah diskvalifikovati projekat. Čak i ako cena projekta nije veća od očekivane dobiti, ona se često poredi sa cenama drugih projekata koji se mogu početi umesto njega, tako da je vreme glavna mera konkurentnosti projekata.

Procene ne utiču isključivo na odluku da li početi projekat, već i na sve ostale aspekte projekta kao što su kvalitet, performanse, prekovremeni rad i slično. Procene napora po kojima je dovoljno manje vremena za razvoj, kao i projektni zadaci koji nisu adekvatno procenjeni dovode do kašnjenja u projektima, prekovremenog rada tima, zahteva za dodatnim budžetom i slično. One takođe mogu da utiču na kvalitet softvera. U slučaju da nema dovoljno vremena ili budžeta, često se ne završavaju aktivnosti kao što su analiza ili testiranje, i svode se na minimum potreban da se kompletira projekat. Međutim, taj „minimum“ napora koji će biti uložen je definisan isključivo vremenom i novcem koji su predviđeni inicijalnim procenama.

Ovo su razlozi zbog kojih je potrebno naći mehanizam koji će dovesti do što je moguće realnijih procena napora, tj. ne previše velikih kako projekat ne bi bio nerealno odbijen, a s druge strane ne previše malih da se ne ugrozi završetak projekta ili kvalitet.

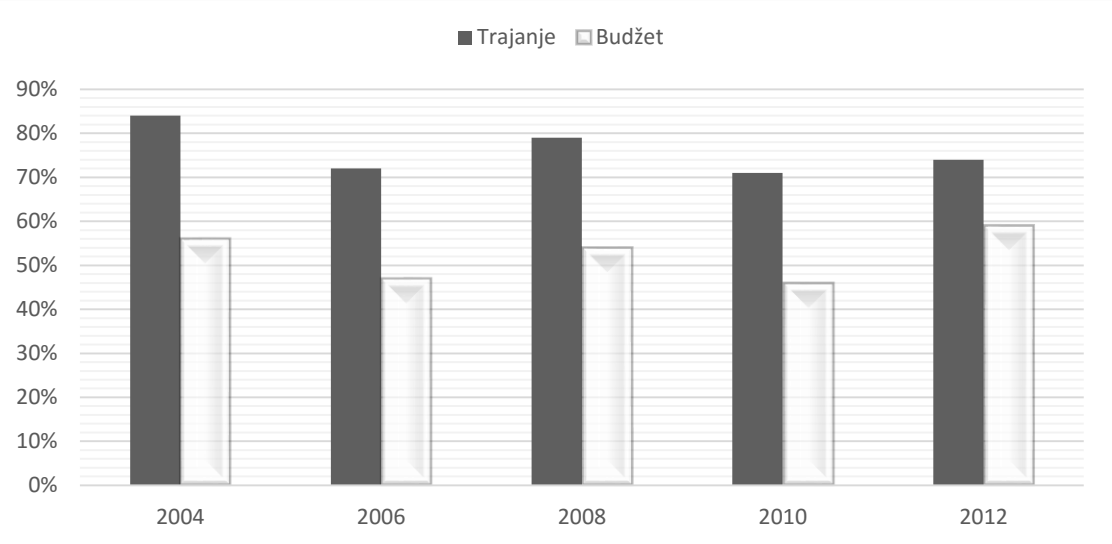
Trenutno stanje u softverskoj industriji po pitanju uspešnosti projekata nije idealno. Standiš grupa (engl. *The Standish Group*) već godinama sakuplja globalne statistike o uspešnim, propalim i diskutabilnim (engl. *challenged*) projektima i objavljuje rezultate svakih par godina [CHAOS, 2013]. U tabeli 1 su prikazani odnosi brojeva uspešnih, propalih i diskutabilnih projekata u poslednjih deset godina [CHAOS, 2013].

Tabela 1. Odnosi uspešnih i neuspešnih projekata.

	2004	2006	2008	2010	2012
Uspešni	29%	35%	32%	37%	39%
Propali	18%	19%	24%	21%	18%
Diskutabilni	53%	46%	44%	42%	43%

Može se primetiti trend u kome broj uspešnih projekata raste, ali procenat propalih projekata je manje-više uvek između 18% i 24%, dok je verovatnoća da će projekat ući u diskutabilnu grupu oko 42% do 44%. Činjenica je da svaki drugi projekat ima velike šanse da uđe u među-zonu gde je diskutabilno da li će uspeti da se završi na vreme ili će propasti.

U propalim i diskutabilnim projektima se u najvećem broju slučajeva probijaju rokovi ili budžet. Na slici 1 [CHAOS, 2013] je prikazano na koliko diskutabilnih projekata dolazi do probijanja planiranih vrednosti trajanja projekta ili budžeta potrebnog da se projekat završi.

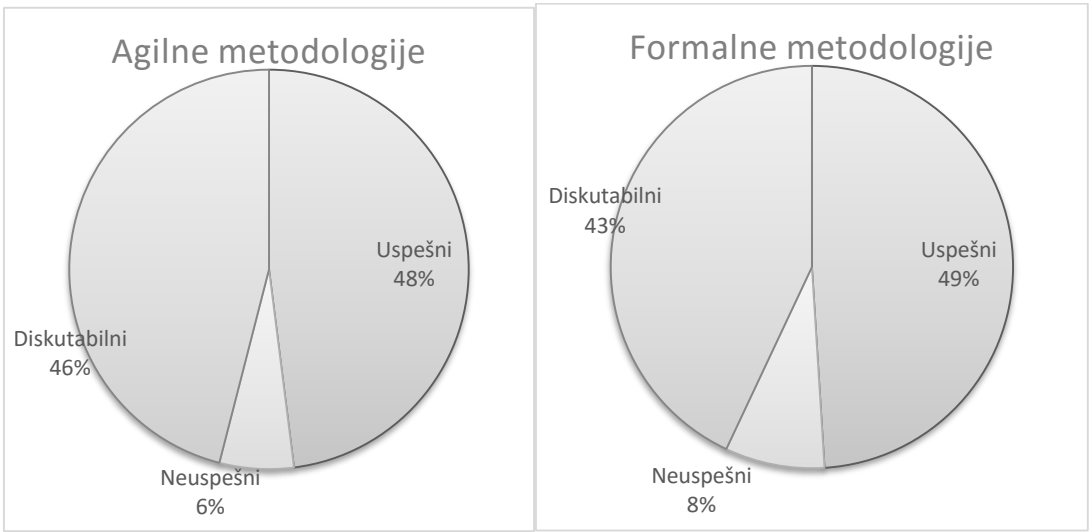


Slika 1. Verovatnoća da se u projektu probije rok ili budžet.

Kao što se može primetiti, probijanje rokova i budžeta je prisutno u 60-75% diskutabilnih projekata. Ovi faktori su direktno uzrokovani lošim procenama napora koji je potrebno uložiti na projektu.

U poslednjih nekoliko godina, trend je podela projekata na manje delove ili iteracije, ograničenje celina koje se rade na kraće rokove (engl. *timeboxing*) [Beck, 1999], [Schwaber and Beedly, 2001], [Kruchten, 2000] kako bi se projekti uspešnije doveli kraju. Ovo je u skladu sa istraživanjima koja pokazuju da je veća verovatnoća propadanja većih projekata [NetoAlvarez, 2003]. Imajući ovo u vidu ima smisla podeliti projekat na manje nezavisne projekte koji će se verovatno uspešnije privesti kraju.

Podela projekta na manje celine se često poistovećuje sa preporukama o primeni agilnih metodologija kako bi se povećala uspešnost projekata. Često se projekti rađeni agilnim metodologijama poistovećuju sa manjim projektima koji se lakše kontrolišu, iako je moguće primeniti formalne metodologije i na manje projekte. CHAOS izveštaj ima i uporedne podatke o odnosima uspešnih i neuspešnih projekata po različitim kategorijama kao što su mali projekti, projektni rađeni agilnim ili formalnim metodologijama i slično. Odnos uspešnosti završetka projekta posmatran u klasi malih projekata u zavisnosti od agilnih ili formalnih metodologija je prikazan na slici 2 [CHAOS, 2013].



Slika 2. Odnosi uspešnih i neuspešnih malih projekata.

Kao što se može primetiti rad na manjim projektima i primene agilnih metodologija ne utiču u proseku previše na uspešnost projekata.

Bez obzira na veličinu i metodologiju, na projektima i dalje ostaje problem procena vremena i napora potrebnih da se implementira projekat.

Ovo je razlog zašto je potrebno istraživati kako se metode procene napora koje se uobičajeno koriste u praksi radi procene napora mogu unaprediti.

1.1. Problemi u proceni napora

U praksi se uglavnom koriste četiri načina za procenu napora: analogija/poređenje, sinteza/dekompozicija, procene stručnjaka i parametarsko/algoritamske metode procene [Popovic, 2010]. Osnovne karakteristike ovih metoda su prikazane u tabeli 2.

Tabela 2. Uporedne karakteristike metoda procena napora.

	Opis	**Prednosti**	**Mane**
Analogija	Poređenje novog projekta sa postojećim sličnim projektima	Procena urađena na osnovu iskustva na realnom uzorku	Ne postoje identični projekti; metoda se svodi na procenu implementacije razlika
Sinteza / dekompozicija	Podela projekta na manje delove koji se nezavisno procenjuju i sabiraju kako bi se dobila procena celog sistema	Tačne procene pošto se manji delovi najlakše procenjuju. Lakša za upotrebu i primenljiva za brze procene	Potrebno je mnogo vremena, često se greši u proceni integracije, preciznost zavisi od nivoa detalja
Procene stručnjaka	Konsultacije sa stručnjacima koji imaju iskustva ili su se već susreli sa sličnim projektima	Nisu potrebni istorijski podaci, lako se može opravdati procena	Teško je razrešiti konflikte među stručnjacima u slučaju razlike u mišljenjima. Stručnost je diskutabilan pojam
Parametarsko algoritamske metode	Procena se vrši određivanjem parametara/mera projekta i primenom algoritama u cilju određivanja vremena	Brzi i laki za korišćenje, maksimalna objektivnost	Zahtevaju istorijske podatke. Tačnost modela i algoritama zavisi od istorijskih podataka

Metodom analogije se procenjuje napor koji je potrebno uložiti na novom projektu tako što se pronađu projekti koji liče na njega. Pretpostavka je da slični projekti zahtevaju približno slične količine napora za izradu, što je neosporno. Međutim, problem je u definisanju kriterijuma kojim se procenjuje sličnost. Pored toga, pošto nikada ne postoje identični projekti, čak i kada se nađu dovoljno slični, potrebno je identifikovati razlike koje se procenjuju nekom drugom metodom. Imajući u vidu da ako su projekti slični, postoji mogućnost da će određene funkcionalnosti iz postojećeg projekta biti iskorišćene, pa metoda analogije može dati i procene napora veće od potrebnih.

Metodom sinteze se procenjuje vreme potrebno da se završe delovi projekta. Delovi projekta mogu biti ili komponente (npr. baza podataka, korisnički interfejs, sistem za

prijavljivanje), neophodni dokumenti (specifikacije, planovi, uputstva za upotrebu) ili projektni zadaci (analiza, testiranje, trening). Vreme koje je potrebno uložiti na projektu se dobija kao suma vremena pojedinačnih komponenti. Ovo je veoma popularna metoda pošto je lakše proceniti vreme potrebno za implementaciju manjih komponenti, a i obim posla se lakše definiše ako se u sistemu implementiraju samo komponente koje su identifikovane. Mana metode je to što identifikacija i analiza svih komponenti iziskuje mnogo vremena, a često se zanemaruje vreme potrebno za njihovu integraciju. Ova metoda je brža od ostalih i prati logički tok misli prilikom analize sistema. Mana joj je to što identifikacija i analiza svih komponenti iziskuje mnogo vremena a često se zanemaruje vreme potrebno za njihovu integraciju. Uz to, tačnost zavisi od nivoa detalja identifikovanih celina, pa je moguće je proglasiti neku funkcionalnu celinu za elementarnu iako se ona može dekomponovati na još elementarnije funkcionalnosti.

Procene stručnjaka [Jorgensen, 2010] se vrše tako što se opis sistema ili delova sistema prosleđuje ljudima sa iskustvom koji daju svoje mišljenje ili projektnom timu koji će implementirati sistem. Metoda se često primenjuje zato što ljudi koji daju procene uvek mogu da stoje iza svojih mišljenja. Mana ove metode je to što je nekada teško uskladiti različita mišljenja stručnjaka. Iako postoje metode kojima se ovo delimično razrešava kao što su Delfi metoda, metoda tri-tačke, triangularna metoda [PMBok, 2004] i slično, mišljenja ipak često nije moguće potpuno usaglasiti. Nije uvek poznato pod kojim pretpostavkama i uslovima su stručnjaci dali svoje procene, pa se one često ne mogu koristiti kao reprezentativni uzorci za ostale metode. Pored toga, može biti diskutabilno ko je relevantan stručnjak i u kojoj meri mu se može verovati.

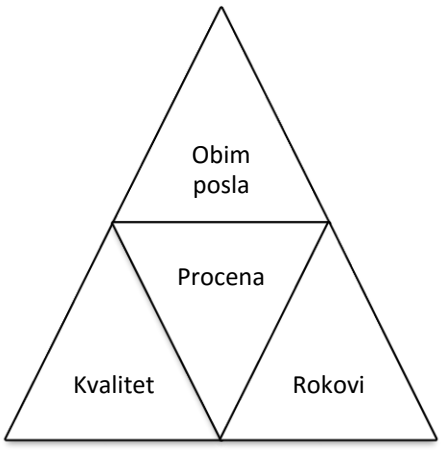
Parametarske/algoritamske metode predstavljaju formalizaciju ostalih metoda. Ideja je da se iskustvo i subjektivna procena zamene funkcionalnim zavisnostima među parametrima i naporom [Popovic, 2010]. S obzirom da se i u ostalim metodama implicitno procenjuju neki parametri sistema, ova metoda predstavlja formalizaciju svih ostalih tehnika. Algoritamske metode se često izbegavaju usled nepostojanja adekvatnih algoritama i modela koji se mogu primeniti, usled čega se prednost daje iskustvu i subjektivnoj proceni. Primena algoritamskih metoda neosporno doprinosi objektivnosti i tačnosti procena. Međutim, za primenu ovih metoda je potrebno imati valjane modele i algoritme koji se mogu primenjivati tokom procene. Na žalost, većina modela za procenu napora

zahteva procenu relativno velikog broja parametara, od kojih značajan deo ne mora imati veze sa projektom čiji se napor procenjuje. Ukoliko bi se definisao odgovarajući model i algoritam po kome bi se eliminisala neophodnost procene nerelevantnih parametara, parametarske metode bi bile značajan dodatak ostalim metodama.

Cilj ovog rada je predstavljanje modela i algoritama koji se mogu primeniti u analizi softvera, kao i kombinovanje različitih metoda ili pravila u postojećim metodama radi poboljšanja procena napora.

1.2. Predmet i cilj istraživanja

Procena napora i resursa potrebnih za realizaciju softverskih projekata predstavlja jedan od najvećih izazova u procesu razvoja softvera [PMBok, 2004]. Napor koji je potrebno uložiti u softverskom projektu zavisi od velikog broja ulaznih parametara, kako funkcionalnih (broja zahteva, složenosti poslovnih procesa i struktura podataka) tako i nefunkcionalnih (zahtevane performanse, iskustvo članova tima, itd.), kao što je prikazano na slici 3.



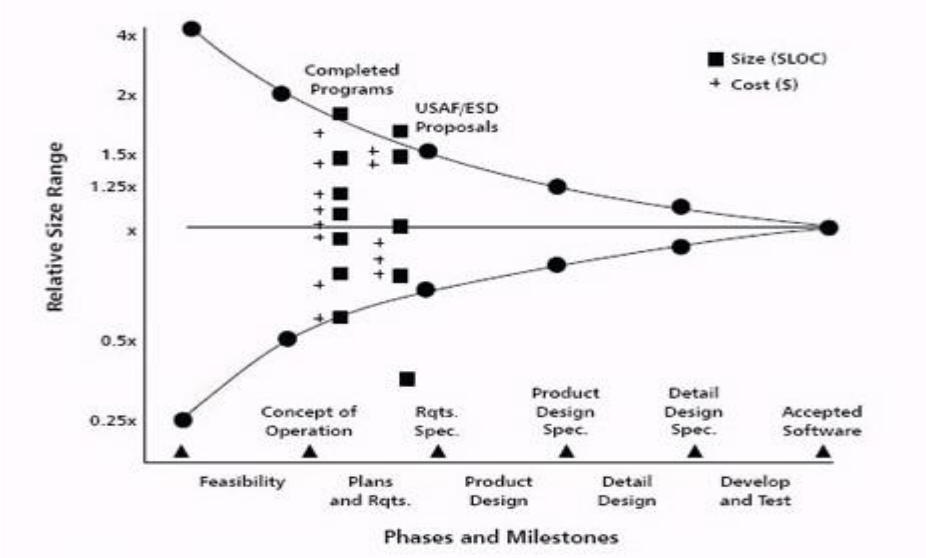
Slika 3. Uticaj različitih faktora na procenu napora na projektu.

Iako je obim posla primarni ulaz u procesu procene napora, ne mogu se zanemariti ni ostali faktori kao što su:

1. Kvalitet – u slučaju da se zahtevaju visoke performanse sistema, kratak odziv, obrada velike količine podataka, lako korišćenje softvera i slično, potrebno je dodatno vreme da bi se ovi zahtevi zadovoljili.

2. Rokovi – često su softverski projekti vezani za neke eksterne rokove kao što su donošenje novih zakona u skladu sa kojima bi trebalo objaviti prve verzije sistema, konferencija na kojima treba objaviti proizvode, konkurencije koja najavljuje da će implementirati sličan proizvod do određenog roka. Činjenica je da napor potreban za implementaciju sistema ne može da se smanjuje proporcionalno broju dodatih članova tima, tako da je potrebno dodati vreme potrebno za komunikaciju i koordinaciju kao i vreme čekanja da se zavisni projektni zadaci završe i sinhronizuju.

Greške procene koje se očekuju u praksi su definisane u radovima dr Berija Boema [Boehm *et al.*, 2000] i instituta za upravljanje projektima (engl. PMI – *Project Management Institute*) [PMBok, 2004]. Po podacima dr Boema, u procesu procene napora se može primetiti da procene konvergiraju ka tačnim vrednostima kako se projekat primiče kraju i kako se otkrivaju nove informacije koje dovode do boljih procena. Ova karakteristika je predstavljena takozvanom kupom nesigurnosti (engl. *Cone of uncertainty*) prikazanom na slici 4 [Boehm *et al.*, 2000].



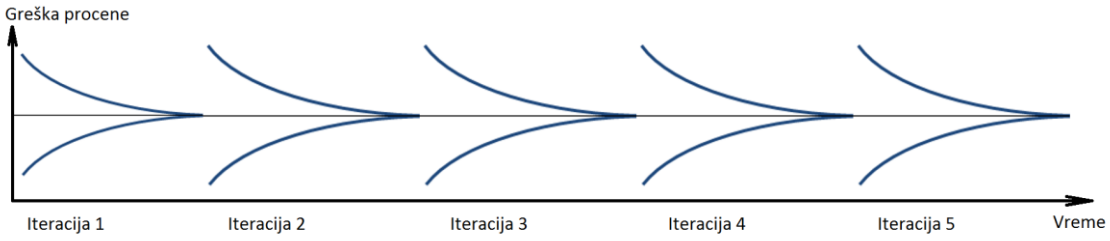
Slika 4. Kupa nesigurnosti procene napora po različitim fazama projekta.

PMI institut ima slične podatke o konvergiranju grešaka. Po PMI kriterijumima, postoje tri vrste procena [PMBok, 2004], [Mulcahy, 2013]:

1. Inicijalne procene (u nastavku obeležene sa PMI-I) koje se daju u trenutku kada je definisan opseg funkcionalnosti koje bi trebalo implementirati, ali još bez detaljne razrade svih slučajeva korišćenja. Greške u ovim procenama variraju u opsegu od -25% do 75%.

2. Budžetske procene (u nastavku obeležene sa PMI-B) koje se daju u trenutku kada je projektni tim dovoljno uveren da je sagledao sve zahteve, definisao tehnologije i arhitekture koje će se koristiti, međutim još uvek ne mora biti razrađen dizajn. U budžetskim procenama se očekuju greške u opsegu od -10% do 25%.
3. Definitivne procene koje se daju kada su potpuno definisani zahtevi, i arhitektura i kada je razrešena većina nepoznatih stvari. Očekivani opseg grešaka je od -5% do 10%.

Može se primetiti da tačnost procene veoma zavisi od faze projekta. Ovo je nepovoljna karakteristika i potrebno je smanjiti ekstremne vrednosti na početku projekata. Iterativne metode pokušavaju da ublaže velike greške procene na početku projekta, tako što dele projekat na manje celine (iteracije, sprintove [Schwaber and Beedly, 2001], itd.) tako da se dobije veliki broj malih kupa nesigurnosti kao što je prikazano na slici 5.



Slika 5. Smanjenje neodređenosti greške procene podelom projekta na iteracije.

Iako ovaj pristup izgleda pogodnije za primenu, treba imati u vidu međusobni uticaj iteracija. Ako se iteracije posmatraju kao mali projekti, oko 10% iteracija bi trebalo da bude neuspešno što uzrokuje ponavljanje iteracija i pomeranje celog projekta. U 40% diskutabilnih iteracija 50% može probiti vreme što opet pomera ostale iteracije. Na ovaj način dobijamo oko 30% iteracija koje će verovatno uzrokovati kašnjenja i probijanja budžeta.

Alternativna metoda bi bila poboljšanje samih metoda procene kako bi se greške procene u kupi nesigurnosti smanjile tokom celog životnog ciklusa projekta što je i predmet istraživanja u ovom radu. Umesto ublažavanja efekta loših procena, potrebno je ispitati kako se postojeće metode procena mogu unaprediti.

Predmet istraživanja je evaluacija postojećih metoda za procenu napora potrebnog da se softverski projekat završi [Jorgensen, 2010], [Trendowicz *et al.*, 2014], [Usman *et al.*, 2014], [Schofield *et al.*. 2013], [Kamal *et al.*, 2011], mogućnost kombinovanja,

prilagođavanja i unapređivanja postojećih metoda u cilju izvođenja novih metoda koje bi dovele do boljih procena.

Unapređenje postojećih metoda promenama pravila za procenu ili primenom metoda veštačke inteligencije je aktuelna tema u ovoj oblasti [Nassiff *et al.*, 2012], [Azzeh, 2013], [Mendes, 2008], [González-Carrasco *et al.*, 2012], [Popovic, Puric, *et al.*, 2015]. Unapređivanje se može postići bilo kombinovanjem parametara i pravila iz postojećih metoda ili primenom naprednih modela za procenu [Han and Kamber, 2006] (npr. višestruka regresija, neuralne mreže [Nassif *et al.*, 2013], i sl.).

Značaj unapređivanja se ogleda u činjenici da je procena napora u aktuelnim uslovima veoma važna aktivnost upravljanja softverskim projektima [CHAOS, 2013]. Procene napora koje su manje od realnih vrednosti često dovode do otkazivanja ili propadanja projekata usled potrošenog budžeta što predstavlja problem kako za softverske kompanije koje su već uložile resurse u projekte tako i za klijente koji na kraju neće dobiti očekivane proizvode ili morati da ulažu dodatna sredstva. S druge strane, procene napora koje su veće od realnih dovode do nekonkurentnosti ponuda za izradu projekata, tako da softverski timovi često ne uspevaju da dobiju i započnu projekat. Unapređenjem postojećih metoda procene bi se ublažili ovi rizici.

1.3. Osnovne hipoteze

Osnovne hipoteze istraživanja su da se postojeće metode procene mogu poboljšati kombinovanjem, izvođenjem novih metoda, primenom naprednih metoda za analizu, itd. Hipoteze koje će biti proverene su:

1. Postojeće metode procene se mogu unaprediti tako što se uzmu u obzir istorijski podaci o projektima koji su već implementirani i za koje su prikupljeni podaci o utrošenim vremenima kao i opisi projekata.
2. Greške procene napora različitih metoda se razlikuju tako da je moguće je odabrati metode koje daju najbolje rezultate procene za određeni skup projekata.
3. Kombinovanjem pravila iz različitih metoda se mogu definisati nove metode u kojima se može proceniti napor sa većom tačnošću od polaznih metoda.
4. Za svaku metodu procene se mogu utvrditi grupe projektnih zadataka za koje se potreban napor može tačnije proceniti. Kombinovanje postojećih metoda za procenu napora tako da svaka proceni samo one delove projekta za koje je

najprimenljivija i kombinovanjem sa metodom sinteze (tj. sabiranjem procena) daje bolje procene napora potrebnog za implementaciju projekta.

5. Primenom metoda veštačke inteligencije se mogu dobiti bolje procene u odnosu na postojeće metode koje uglavnom koriste linearne zavisnosti između veličine i procenjenog napora.
6. Različite metode se mogu primeniti u različitim fazama projekta, tako da se mogu identifikovati optimalne metode procene napora po svakoj fazi projekta. Na ovaj način se može definisati postupak kojim se procene daju tokom životnog ciklusa projekta i konvergiraju ka tačnim vrednostima.

Pod pretpostavkom da su ove hipoteze tačne, moguće je definisati metode koje mogu unaprediti postojeće tehnike procene napora.

1.4. Metode istraživanja

Prvi korak u istraživanju je definicija procesa, metoda i algoritama za procenu uticaja funkcionalnih zahteva i ostalih faktora koji utiču na napor koji je potrebno uložiti na projektu.

U okviru istraživanja će biti izvršena evaluacija postojećih metoda procene na realnom skupu softverskih projekata tako što će se odrediti greška procene svake metode. Utvrdiće se i potencijalni uzroci loših procena metoda tako što će se identifikovati grupe projektnih zadataka za koje pojedine metode daju procene napora sa velikim greškama. U cilju komparativne evaluacije metoda po projektima i grupama projektnih zadataka biće korišćena multi-dimenziona analiza pomoću analitičkih kocki (engl. *On-Line Analytical Processing cubes*) [MDX, 2012] u kojima će se procenjene i stvarne vrednosti napora porediti po dimenzijama (veličinama dobijenim različitim metodama, projektima, i grupama projektnih zadataka), kao i R jezik za statističku analizu [R, 2014].

Analiza obuhvata i klasifikaciju metoda na osnovu faze projekta kada se mogu primeniti (npr. metode koje se mogu primeniti u ranoj fazi projekta, tokom analize, po definisanju dizajna i slično). Klasifikacija metoda je potrebna kako bi se identifikovale metode koje je moguće primenjivati u različitim fazama projekta kao i očekivane greške procene napora po fazama.

Ključni deo istraživanja predstavlja izvođenje analitičkog modela za kombinovanje pravila procene uticaja funkcionalnih i nefunkcionalnih parametara softvera u postojećim metodama u cilju izvođenja novih poboljšanih metoda procene napora.

1.5. Očekivani naučni doprinos

Pre svega, u ovoj disertaciji će biti dat pregled postojećih metoda koje se koriste radi procene napora na projektima kao i njihova komparativna evaluacija. Glavni naučni doprinos ovog istraživanja predstavlja skup novih originalnih metoda za procenu napora potrebnog za implementaciju softverskih sistema. Nove metode će biti izvedene iz postojećih metoda uz primenu sledećih modifikacija:

1. Izvođenje analitičkog modela za kombinovanje pravila za procenu uticaja funkcionalnih i nefunkcionalnih zahteva iz različitih metoda,
2. Analiza tačnosti procene postojećih metoda sa aspekta procene napora grupa projektnih zadataka (npr. analiza, dizajn, programiranje, testiranje, upravljanje projektom). Ova vrsta analize nije prisutna u postojećoj literaturi
3. Kombinovanje ekspertskih i formalnih metoda procene. Ova vrsta analize nije prisutna u postojećoj literaturi.
4. Primene metoda veštačke inteligencije (kao što su neuralne mreže [Han and Kamber, 2006]) koje se uveliko koriste u ostalim naučnim i inženjerskim oblastima radi procene napora na projektima.

Greške procene ovakvih metoda bi trebalo da budu manje od grešaka procene originalnih metoda pošto se uklanjaju slabosti koje uzrokuju povećanje grešaka procene u originalnim metodama.

1.6. Struktura rada

Najpre će biti data klasifikacija i komparativna analiza postojećih metoda za procenu veličine softvera radi primene u procesu procene napora na projektima. U zavisnosti od parametara i dokumenata koji su na raspolaganju u projektima i fazama kada se dokumenti mogu koristiti, metode će biti klasifikovane kao metode koje se mogu primeniti u ranim fazama, tokom elaboracije/analize zahteva i tokom dizajna. Za svaku fazu će biti definisane očekivane greške procene koje će se koristiti kao kriterijum za komparativnu evaluaciju metoda.

Odabrane metode za procenu veličine softvera će se primeniti na prikupljene projekte kako bi se napravila baza projekata sa podacima o veličinama i uloženom naporu za svaki projekat.

U istraživanju će biti analizirane i identifikovane prednosti i slabosti postojećih metoda kako bi se utvrdilo kako se mogu kombinovati metode i kako se mogu poboljšati procene napora.

Bitan deo istraživanja je analiza mogućnosti kombinovanja postojećih metoda kako bi se dobili još kvalitetniji predikcioni modeli. Pravila iz postojećih metoda će biti kombinovana kako bi se napravile nove metode čije će tačnosti procene napora biti upoređene sa originalnim metodama.

U cilju formalne validacije rezultata koristiće se statistički parametri za ocenu kvaliteta grešaka procena. Kao rezultat istraživanja će biti predstavljene nove metode koje se mogu primenjivati radi procene napora u softverskim projektima.

Struktura doktorske disertacije se može prikazati kroz sledeće dve celine:

1. U prvom delu disertacije će biti predstavljene metode koje su korišćene u analizi, matematički modeli i alati koji se koriste radi obrade podataka. Pored toga biće predstavljene i postojeće metode za ocenjivanje veličine softvera, uticaja nefunkcionalnih zahteva i procene napora.
2. U drugom delu disertacije će biti upoređene postojeće metode procene napora i predložena njihova unapređenja. Drugi deo sadrži sledeće celine:
    a. Komparativna analiza postojećih metoda za procenu veličine softverskih projekata i njihove primene radi procene napora na projektima.
    b. Kombinovanje pravila za procenu veličine softvera i uticaja nefunkcionalnih zahteva u cilju unapređenja procena napora.
    c. Analiza mogućnosti primene postojećih metoda radi procene napora potrebnog da se implementiraju pojedini projektni zadaci.
    d. Kombinovanje formalnih i subjektivnih metoda, kao i metoda veštačke inteligencije u cilju unapređenja procena napora.

I Deo

U prvom delu rada je predstavljeno trenutno stanje u oblati procene napora na softverskim projektima. U narednom poglavlju će biti predstavljeni matematički modeli i metode koji se trenutno koriste u analizi, kao i metode za skladištenje i predstavljanje rezultata analize. Pored toga, u ovom delu teze će biti predstavljene i metode koje se koriste kako bi se ocenila veličina softverskih sistema na osnovu programskog koda i funkcionalnosti, metode kojima se procenjuje uticaj nefunkcionalnih zahteva, kao i metode kojima se na osnovu veličina softverskih projekata i uticaja nefunkcionalnih zahteva procenjuje napor potreban da se implementira softverski projekat.

2. Metode koje se koriste radi procene napora na projektima

U ovom poglavlju će biti predstavljena metodologija, tehnike i alati koji će biti korišćeni radi analize. Poglavlje obuhvata sledeće celine:

1. Definiciju mera koje će biti korišćene radi kvantifikovanja sistema i na osnovu kojih će se vršiti procena napora.
2. Metode kojima se procenjuje uticaj funkcionalnosti, nefunkcionalnih zahteva, i svih ostalih faktora koji utiču na napor koji je potrebno uložiti na projektu.
3. Modeli korišćeni za skladištenje podataka.
4. Procedure za analizu rezultata.
5. Alati i jezici korišćeni za analizu podataka.

2.1. Definicija mera

Numeričke karakteristike (ili mere) omogućavaju da se različiti sistemi opišu parametrima kako bi se mogli analizirati i upoređivati. U softverskim sistemima se može koristiti veliki broj mera kao što su softverske mere koje direktno opisuju veličinu pojedinih komponenata sistema (npr. broj linija koda, klasa, strana dokumentacije), sistemske mere koje opisuju softver u njegovom okruženju (npr. performanse, prenosivost na druge platforme), procesne mere koje karakterizuju efikasnost tima i performanse razvoja, korisničke mere koje predstavljaju ocenu sistema sa stanovišta korisnosti, stabilnosti, kvaliteta proizvoda, zadovoljstva klijenta i slično.

Mere se mogu podeliti na dve osnovne vrste – direktne i izvedene mere [CMMI, 2012]. Direktne mere se određuju direktnim uvidom u dokumentaciju, bez dodatne obrade podataka. Za izvedene mere je potrebno vršiti dodatnu analizu ili transformaciju kako bi se odredile na osnovu direktnih mera.

Po tipu, mere mogu biti skalarne ili vektorske [Popovic, 2010], [CMMI, 2012]. Skalarne mere se izražavaju numerički u jedinici koja je pridružena meri. Vektorske mere predstavljaju uređenu torku prostih logički povezanih mera, koje zajedno predstavljaju korisne informacije o sistemu.

2.1.1. Direktne mere

Direktne ili osnovne mere [CMMI, 2012] predstavljaju parametre sistema koji se prebrojavaju bez dodatne obrade ili analize. One se utvrđuju direktnim uvidom u dokumentaciju ili direktnim intervjuima sa članovima tima koji poseduju određene informacije. Primeri direktnih mera mogu biti vreme potrošeno na projektu ili pojedinim projektnim zadacima, broj slučajeva korišćenja i slično. Vreme i broj slučajeva korišćenja su primeri skalarnih mera.

Neke mere ne mogu nezavisno pružiti korisne informacije o karakteristikama sistema tako da se moraju kombinovati sa drugim merama koje su u tesnoj vezi sa njima kako bi se izveo skup korisnih informacija. Ovakve parcijalne mere koje se koriste isključivo za dalju obradu se nazivaju dimenzije. Skup dimenzija koje su u međusobnoj vezi predstavljaju vektorske mere. Primeri vektorskih direktnih mera su:

1. Funkcionalna veličina sistema određena po NESMA indikativnoj metodi [Abran *et al.*, 1999] koja je predstavljena kao dvodimenzionalni vektor brojeva internih i eksternih fajlova (Ni, Ne).
2. Funkcionalna veličina sistema određena po Mark II metodi [Symons, 1991] koja je predstavljena kao trodimenzionalni vektor broja ulaza, izlaza i objekata (Ni, Ne, No).

I za skalarne i za vektorske mere važi pravilo da se određuju direktnim uvidom u dokumentaciju bez dodatne obrade.

2.1.2. Izvedene mere

Izvedene mere zahtevaju dodatnu analizu pre nego što se prikupe. Direktne mere često daju previše detalja iz kojih je teško izvesti zaključke tako da je potrebno uraditi neku transformaciju kao što su [Popovic, 2010]:

1. Klasifikacija – umesto informacija o veličinama objekata kao što je broj strana, akcija u procesima, podataka u fajlovima, često je pogodnije klasifikovati objekte kao proste, srednje i složene kako bi se pojednostavila analiza,
2. Određivanje norme – iako vektorske mere precizno određuju veličine sistema, ipak je potrebno odrediti neku skalarnu veličinu koja bi omogućila lakše poređenje mera.

Klasifikacija se koristi u različitim metodama za određivanje veličine sistema. U metodi tačaka slučajeva korišćenja [Karner, 1993] se vrši klasifikacija slučajeva korišćenja na

jednostavne (L), srednje (A) i komplikovane (C) slučajeve korišćenja na osnovu broja transakcija u scenarijima slučajeva korišćenja i na osnovu njih se definiše vektorska mera koja predstavlja veličinu sistema u vidu vektora UC = (UC_L, UC_A, UC_C). U metodi funkcionalnih tačaka [IFPUG, 1994], [Nesma, 1997] se vrši klasifikacija funkcionalnosti i struktura podataka na jednostavne (L), srednje (A) i komplikovane (C) kako bi se dobio petnaesto-dimenzionalni vektor veličine sistema.

2.1.3. Operacije nad merama

Potrebno je definisati operacije koje se mogu vršiti nad merama što je veoma važno radi analize rezultata. Nad skalarnim merama se primenjuju standardne matematičke operacije, ali se za vektorske mere moraju definisati posebne operacije koje se koriste prilikom obrade rezultata. Najčešće korišćena operacija je množenje vektora skalarom, gde skalarna veličina α (tipično realan broj) množi vektor $x = (x_1, x_2, ..., x_n)$, čime se dobija novi vektor sa dimenzijama $(\alpha x_1, \alpha x_2, ..., \alpha x_n)$ [Cvetkovic *et al.*, 2004].

Pored toga, bitna operacija na vektorima je određivanje norme vektora [Cvetkovic *et al.*, 2004]. Norma vektorskog prostora V, u oznaci $\|x\|$, je bilo koja funkcija $V \rightarrow R_0^+$ koja vrši preslikavanje vektorskog prostora V u skup ne-negativnih realnih brojeva R_0^+ sa sledećim osobinama [Popovic, 2010]:

$$\|x\| = 0 \Leftrightarrow x = 0 \quad (1)$$

$$\|\alpha x\| = \alpha \|x\| \quad (2)$$

$$\|x + y\| \leq \|x\| + \|y\| \quad (3)$$

Norma se može poistovetiti sa intenzitetom vektora i predstavlja najčešće korišćenu operaciju za konvertovanje vektorskih mera u skalarne. U matematici [Cvetkovic *et al.*, 2004] je već definisano dosta funkcija koje se mogu koristiti kao norme vektorskog prostora kao što su:

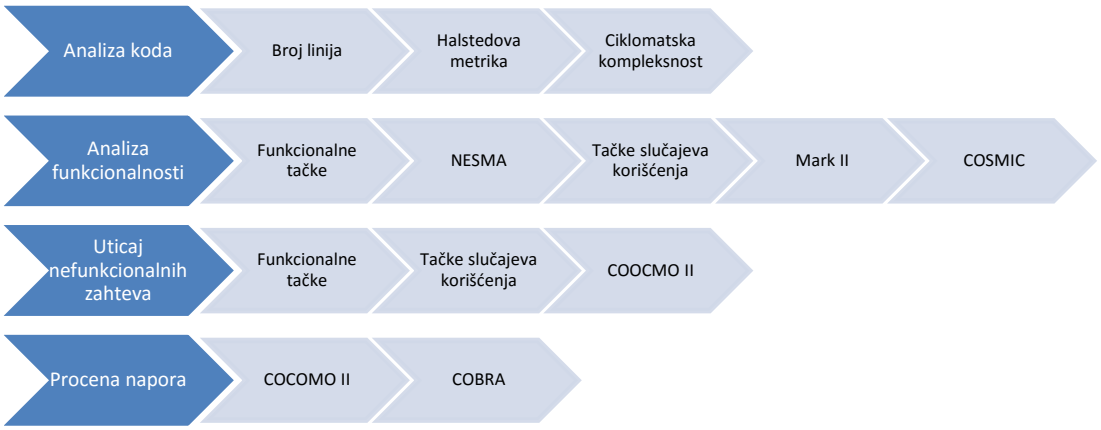
1. Euklidova norma – $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$ koja predstavlja koren sume kvadrata dimenzija je najčešće korišćena norma u matematici.

2. Menhetn norma – $\|x\| = \sum_{i=1}^{N} |x_i|$ gde se apsolutne vrednosti dimenzija sabiraju, koja se koristi u COSMIC-FFP metodi [Abran *et al.*, 1999].

3. Linearna Menhetn norma – $\|x\| = \sum_{i=1}^{N} k_i * |x_i|$ u kojoj se uvode težinski koeficijenti koji množe apsolutne vrednosti dimenzija. U slučaju da su sve dimenzije vektora ne-negativne ova norma je ekvivalentna skalarnom proizvodu vektora sa vektorom težina (k_1, k_2,... k_{n-1}, k_n). Ova norma se primenjuje u svim važnijim metodama određivanja funkcionalne veličine sistema kao što su funkcionalne tačke [IFPUG, 1994], tačke slučajeva korišćenja [Karner, 1993] i slično.

2.2. Metode za merenje i procenu veličine softvera

Trenutno postoji veliki broj metoda kojima se softver kvantifikuje sa aspekta veličine ili kompleksnosti. U softveru se mogu kvantifikovati različiti parametri kao što su funkcionalnosti, ali i uticaj nefunkcionalnih zahteva. Postoji i posebna grupa metoda kojima se kao rezultat direktno dobija vreme koji je potrebno uložiti na projektu. Pregled metoda koje se koriste u praksi je prikazan na slici 6.



Slika 6. Metode procene veličine i napora.

Metode se grubo mogu podeliti na:

1. Metode kojima se kvantifikuje opseg posla. Ove metode daju numeričke veličine koje bi trebale da budu srazmerne obimu posla koji je potrebno uraditi. Numeričke vrednosti se dobijaju analizom programskog koda ili funkcionalnosti.

2. Metode za procenu uticaja nefunkcionalnih zahteva. Veliki broj nefunkcionalnih zahteva utiče na napor koji je potrebno uložiti na projektu. Veliki broj metoda za

kvantifikovanje veličine softvera uključuje i neka pravila kojima se kvantifikuje uticaj ovih zahteva.

3. Metode za procenu napora na osnovu veličine. Ove metode imaju definisana pravila i modele na osnovu kojih se može proceniti napor koji je potrebno uložiti kako bi se završio projekat.

Metode za kvantifikovanje projekata na osnovu funkcionalnosti su detaljnije opisane u poglavlju 3, dok su metode za kvantifikovanje uticaja nefunkcionalnih zahteva opisane u poglavlju 4.

2.3. Procedure za analizu rezultata

Da bi se analizirali prikupljeni podaci potrebno je definisati procedure za analizu. Bez definicije ovih procedura postojala bi mogućnost haotičnog i neorganizovanog pristupa analizi podataka, što bi moglo da dovede do prikupljanja nekorektnih mera i izvlačenja pogrešnih zaključaka. U ovoj sekciji su objašnjene metode koje su korišćene radi analize, utvrđivanja preciznosti procena i statističke validacije rezultata, kao i alati korišćeni za analizu.

2.3.1. Regresija

Regresija je metoda kojom se na osnovu skupa nezavisnih i zavisnih vrednosti određuje funkcionalna zavisnost kojom se zavisne promenljive mogu izraziti u funkciji od nezavisnih [Harrell, 2001], [Popovic, 2010]. Regresija omogućava da se na osnovu skupa parova nezavisnih i zavisnih promenljivih (x_i, y_i) odredi funkcija $\hat{y}(x)$ kojom se na osnovu promenljive x_i može odrediti procenjena vrednost $\hat{y}_i$ koja će biti dovoljno blizu stvarnoj vrednosti zavisne promenljive y_i, kao što je prikazano u formuli (4).

$$y_i = \hat{y}_i(x_i) + \varepsilon_i \tag{4}$$

Stvarna vrednost y_i se predstavlja kao zbir procenjene vrednosti $\hat{y}_i$ i greške procene ε_i. Regresioni modeli se kreiraju tako da se minimizuju greške u formuli (4).

U ovom radu procenjena vrednost je vreme koje je potrebno uložiti na projektu, dok su nezavisne promenljive različite mere kojima se procenjuje uticaj količine koda, obima funkcionalnosti, nefunkcionalnih zahteva, i slično.

Prilikom kreiranja regresionih modela potrebno je pretpostaviti funkcionalnu zavisnost među promenljivama – primeri funkcionalnih zavisnosti mogu biti linearna (5), polinomska (6), logaritamska (7), eksponencijalna (8), [Popovic, 2010] itd.

$$\hat{y}_i = \beta_0 + \beta_1 * x_i \quad (5)$$

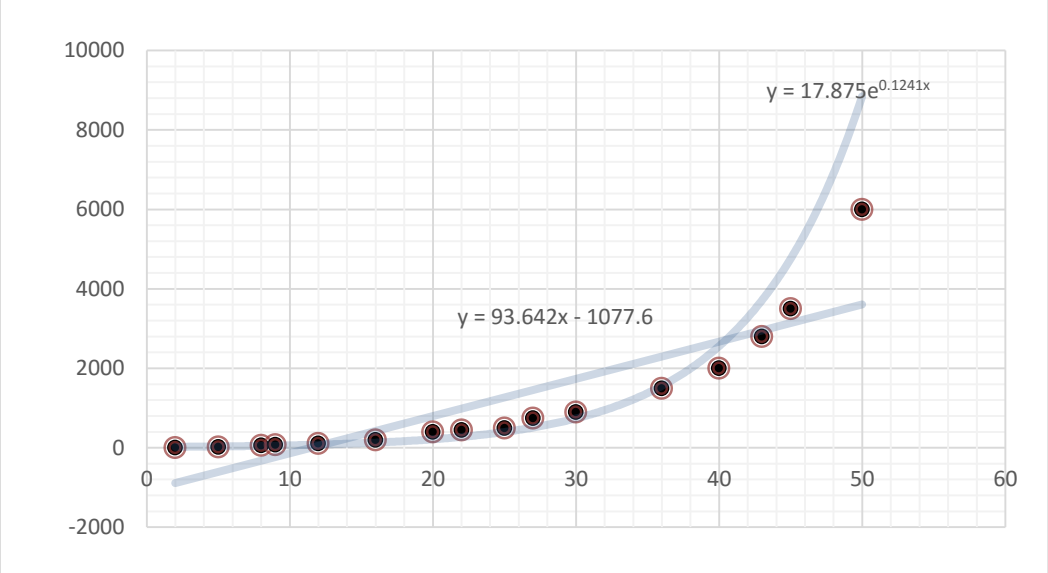
$$\hat{y}_i = \beta_0 + \beta_1 * x_i + \beta_2 * x_i^{\,2} \quad (6)$$

$$\hat{y}_i = \beta_0 + \beta_1 \ln(x_i) \quad (7)$$

$$\hat{y}_i = \beta_0 + \beta_1 * e^{x_i} \quad (8)$$

Parametri β_i se određuju tako da se minimizuje greške procene ε_i iz formule (4). Najčešće se minimizuje srednje kvadratna greška (tj. zbir kvadrata pojedinih grešaka).

Na slici 7 je prikazana primena linearne i eksponencijalne regresije radi određivanja funkcionalne zavisnosti između nezavisnih i zavisnih promenljivih.



Slika 7. Primena linearne i eksponencijalne regresije nad istim skupom podataka.

Na slici 7 se može videti da bi polinomska ili eksponencijalna zavisnost mogla da bude dobar izbor za posmatrani skup podataka. Vizuelna ocena može biti dobar indikator koje zavisnosti bi trebalo koristiti u regresiji. Ako to nije moguće mogu se isprobati različite funkcionalne zavisnosti kako bi se našla ona koja daje najmanju grešku.

U radu je korišćenja linearna regresija u kojoj je skup zavisnih promenljivih prikazan kao zakonitost y = k*x + n, gde su k i n parametri linearne zavisnosti, nad skupom od N parova podataka (x_i, y_i) tako da se dobije minimalna srednje kvadratna greška. Ova greška zavisi od parametara k i n prikazanih u formuli (9) [Popovic, 2010].

$$E(k,n) = \sum (\hat{y}_i - y_i)^2 = \sum (kx_i + n - y_i)^2 \tag{9}$$

Minimizacijom funkcije iz formule (9) po parametrima k i n se dobijaju optimalne vrednosti za parametre linearne regresije. Koeficijenti koji opisuju linearnu zavisnost se mogu odrediti prema formulama (10) i (11) [Popovic, 2010].

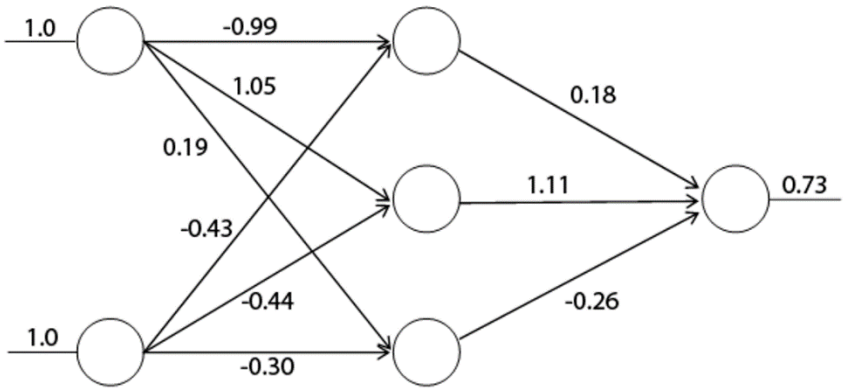
$$k = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{N\sum x_i^2 - \left(\sum x_i\right)^2} \tag{10}$$

$$n = \frac{\sum y_i - k\sum x_i}{N} \tag{11}$$

U nastavku će biti pokazano da većina alata za analizu podataka ima ugrađene funkcije za računanje ovih koeficijenata.

2.3.2. Neuralne mreže

Neuralne mreže [Han and Kamber, 2006], [Jin *et al.*, 2012] su popularan metod za kreiranje predikcionih modela. Ideja je da se predikcija vrši tako što se informacije prosleđuju do elemenata (ili neurona u terminologiji neuralnih mreža) koji ih modifikuju i prosleđuju dalje drugim elementima. Informacije koje ulaze u mrežu predstavljaju nezavisne promenljive na osnovu kojih se vrši predikcija (u ovom slučaju veličine projekata), dok su informacije koje izlaze iz mreže zavisne promenljive (u ovom slučaju procene napora na projektima). Primer jedne neuralne mreže je prikazan na slici 8.



Slika 8. Primer neuralne mreže sa dva ulaza i jednim internim slojem.

Na slici 8 je prikazana neuralna mreža sa dva ulaza, jednim internim slojem neurona i jednim izlazom. Vrednosti na ulazu se množe koeficijentima i prosleđuju internom sloju gde se sabiraju. Rezultati se ponovo množe novim koeficijentima i prosleđuju sledećem sloju (u ovom slučaju izlazu). Ovo je jednostavna mreža ali u praksi se mogu koristiti dosta složenije mreže koje imaju više internih slojeva, povratne veze od izlaza i internih slojeva ka slojevima bližim ulazu i slično.

Koeficijenti u mreži se dobijaju treniranjem mreže – mreži se prosleđuje niz ulaznih podataka kao i niz očekivanih vrednosti tako da mreža može da nađe optimalne koeficijente kojima se na osnovu ulaza određuje izlaz.

Prednost neuralnih mreža u odnosu na linearnu regresiju sa više promenljivih leži u činjenici da nije potrebno pretpostaviti funkcionalnu zavisnost pošto će je mreža sama otkriti u fazi treniranja. Mana leži u činjenici da je potrebno dosta podataka kako bi se mreža dobro trenirala.

2.3.3. Greške procene

Greške procene govore koliko je predikcioni model tačno predvideo zavisnu promenljivu. Postoji veliki broj kriterijuma kojima se predstavljaju greške merenja [Conte *et al.*, 1986] kao što su reziduali (engl. *residuals*) koji predstavljaju razliku između stvarne i procenjene vrednosti prikazani u formuli (12), relativna greška MRE_i (*eng. Magnitude of Relative Error*) prikazana u formuli (13) i Z odnos [Foss *et al.*, 2003], [Kitchenham *et al.*, 2001] prikazan u formuli (14) koji predstavlja odnos procenjene i stvarne vrednosti.

$$residual_i = y_i - \hat{y}_i \tag{12}$$

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{13}$$

$$Z_i = \frac{\hat{y}_i}{y_i} \tag{14}$$

Ove veličine se koriste kao procena greške procene pojedinačnih projekata. Sumarni parametri koji daju informaciju o kvalitetu samog predikcionog modela su srednja vrednost relativne greške (engl. *Mean Magnitude of Relative Error* – MMRE) i procenat relativnih grešaka manjih od 25% - PRED(25), prikazane u formulama (15) i (16) [Conte *et al.*, 1986]. U formuli (16) COUNTALL je ukupan broj projekata nad kojima je izvršena analiza.

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i \tag{15}$$

$$\text{PRED(25)} = \text{COUNT(proj)/COUNTALL, MRE(proj)} < 25\% \tag{16}$$

2.3.4. Statističke metrike

Korišćenje isključivo grešaka opisanih u prethodnoj sekciji, bez korišćenja nekih dodatnih metoda za validaciju modela, je često osporavano u literaturi [Kitchenham *et al.*, 2001], [Foss *et al.*, 2003], [Shepperd and MacDonell, 2012] kao dovoljan kriterijum za procenu kvaliteta predikcionih modela. Zbog toga se u praksi koristi veliki broj statistika kojima se opisuju karakteristike skupova podataka. Osnovne karakteristike skupova podataka su srednja vrednost i standardna devijacija koji se određuju na osnovu formula (17) i (18) [Merkle i Vasic, 1997].

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{17}$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{18}$$

Srednja vrednost, kao što joj ime kaže, predstavlja aritmetičku sredinu skupa podataka dok standardna devijacija predstavlja očekivanje odstupanja ostalih vrednosti u skupu od srednje vrednosti. Pored srednje vrednosti i standardne devijacije distribucija podataka se opisuje kvantilima [Kutner *et al.*, 2004]. Kvantil reda k (gde je k broj između 0 i 1)

predstavlja onaj član skupa koji je veći od 100*k % članova skupa. Pored minimuma i maksimuma (kvantili nultog i prvog reda), koriste se i prvi kvartil (kvantil 0.25 reda) koji predstavlja onaj element u skupu gde je jedna četvrtina elemenata skupa manja od njega dok su tri četvrtine veće), medijana – kvantil 0.5 reda koji predstavlja onaj element koji ima jednak broj elemenata koji su veći i manji od njega, i treći kvartil (kvantil 0.75 reda) koji predstavlja element od koga su tri četvrtine elemenata skupa manje a jedna četvrtina veća.

Srednja vrednost, standardna devijacija i kvartili daju informaciju o distribuciji podataka. Pored karakterizacije skupa podataka često je potrebno porediti zavisnost podataka između dva skupa podataka korišćenjem koeficijenta korelacije među skupovima. Koeficijent korelacije je parametar koji služi da uporedi veze među vektorima ili nizovima i određuje se na osnovu formule (19) [Popovic, 2012].

$$Correl(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{19}$$

U radu je koeficijent korelacije korišćen radi utvrđivanja veze između skupa veličina sistema određenim raznim metodama sa veličinom sistema koja predstavlja napor koji treba da bude uložen radi implementacije.

Pored koeficijenta korelacije, često se koristi koeficijent determinisanosti u oznaci R^2 koji predstavlja kvadrat koeficijenta korelacije [Nagelkerke, 1991]. Koeficijent R^2 predstavlja procenat projekata koji se mogu objasniti predikcionim modelom i pored MMRE i PRED(25) parametara predstavlja još jedan parametar kvaliteta predikcionih modela.

S obzirom da je osnovna tema rada unapređenje metoda procene napora, bitno je identifikovati statističke testove za procenu boljih predikcionih modela. Akaike informacioni kriterijum (AIC) je mera kvaliteta statističkih modela za dati niz podataka [Sakamoto *et al.*, 1986], tako da može da predstavlja kriterijum za selekciju boljeg predikcionog modela. Pored AIC kriterijuma, Wilkoksonov označeni rang test [Venables

and Ripley, 2002] testira verovatnoću da su greške procene u jednom modelu manje od grešaka u drugom.

2.3.5. Unakrsna validacija rezultata

Radi dobijanja objektivnih rezultata, potrebno je odvojiti skup projekata koji se koriste radi kreiranja predikcionih modela od onih koji se koriste za evaluaciju. Primena istih podataka tokom kreiranja predikcionih modela i procene greške može dovesti do neželjenih korelacija među rezultatima. Zbog toga se koriste metode unakrsne validacije (engl. *cross-validation*) u kojima se skup podataka deli na trening skup koji se koristi za kreiranje predikcionih modela i validacioni skup koji se prosleđuje predikcionom modelu i nad kojim se utvrđuju greške procene i tačnost modela [Kocaguneli and Mensies, 2013].

Postoji veliki broj metoda kojima se definiše na koji način se može podeliti skup projekata na trening i validacioni skup. U istraživanju je korišćena unakrsna validacija izbacivanjem jednog elementa (engl. LOOCV – *Leave one out cross validation*) [Kocaguneli and Mensies, 2013]. LOOCV algoritam se može opisati sledećim koracima:

1. Iz skupa projekata koji se analiziraju se bira jedan projekat.
2. Predikcioni model se kreira na osnovu podataka iz ostalih projekata.
3. Greška se utvrđuje primenom predikcionog modela na izabrani projekata koji nije učestvovao u procesu kreiranja modela.
4. Postupak se ponavlja dok se svaki projekat ne odabere kao validacioni projekat jednom i samo jednom.

Na ovaj način se dobija veliki broj nezavisnih iteracija u kojima se mogu prikupiti nezavisni podaci o greškama. Detaljniji opis prednosti LOOCV metoda u poređenju sa ostalima je objašnjen u [Kocaguneli and Mensies, 2013].

2.3.6. Evaluacija validnosti predikcionih modela

Bitan deo analize je provera da li je predikcioni model validan. Činjenica da model daje male greške ne znači da je dobar na drugom skupu projekata. Kada se prikupe greške merenja dobijene uporednom validacijom potrebno je proveriti da li im je distribucija očekivana. Osnovni parametar na osnovu kojeg se može odbaciti predikcioni model je provera koeficijenta korelacije i p-vrednosti modela [Nuzzo, 2014], [Schervish, 1996].

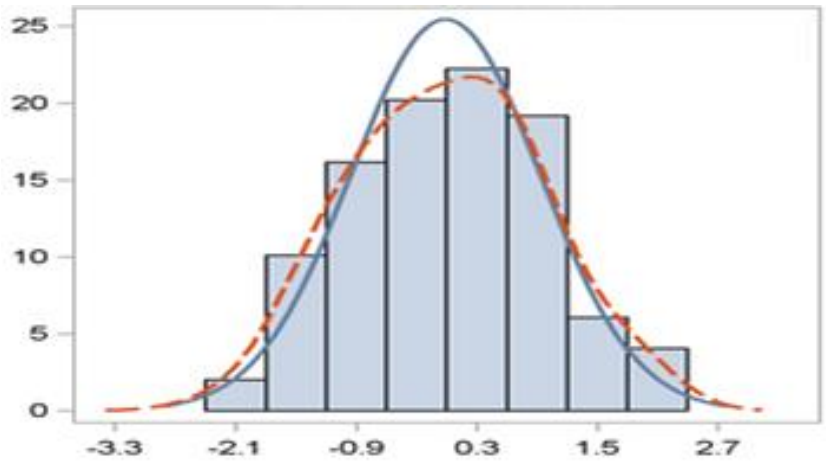
Na osnovu ova dva parametra se utvrđuje da li su nezavisne i zavisne promenljive korelisane – ako nisu onda nema smisla koristiti regresioni model.

Pored ovih osnovnih provera, potrebno je proveriti normalnu distribuciju grešaka, homoskedastičnost i nezavisnost grešaka [Kutner *et al.*, 2004].

Svaki predikcioni model koji ne prođe testove se odbacuje i zaključuje se da se napor ne može proceniti primenom linearne regresije na osnovu veličine projekta.

2.3.6.1. Normalna distribucija grešaka

Raspodela grešaka procene (reziduala) mora da bude u skladu sa normalnom raspodelom [Merkle i Vasic, 1997]. Na slici 9 je prikazana raspodela grešaka koja nije u skladu sa normalnom raspodelom.



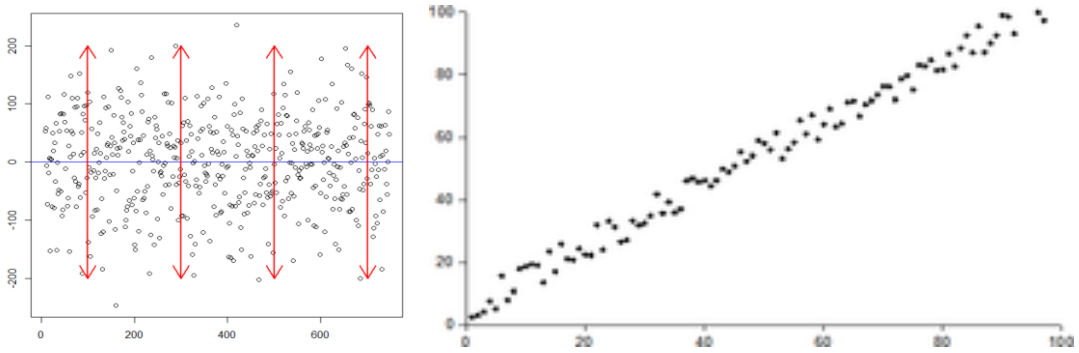
Slika 9. Raspodela grešaka koja ne odgovara normalnoj raspodeli.

U slučaju da srednja vrednost grešaka nije nula, predikcioni model unosi konstantnu grešku procene u svakoj proceni. U slučaju da greške nisu ravnomerno raspodeljene sa obe strane normalne raspodele, predikcioni model ili procenjuje veće ili manje vrednosti u većini slučaja.

2.3.6.2. Homoskedastičnost

Greške ne treba da zavise od nezavisne ili zavisne promenljive [Kutner *et al.*, 2004]. Loši predikcioni modeli imaju male greške na malim vrednostima a veće na većim. U takvim predikcionim modelima se može primetiti divergiranje procena u zavisnosti od nezavisnih

promenljivih. Idealni predikcioni modeli bi morali da imaju greške u istim granicama na celom domenu, kao što je prikazano na slici 10.



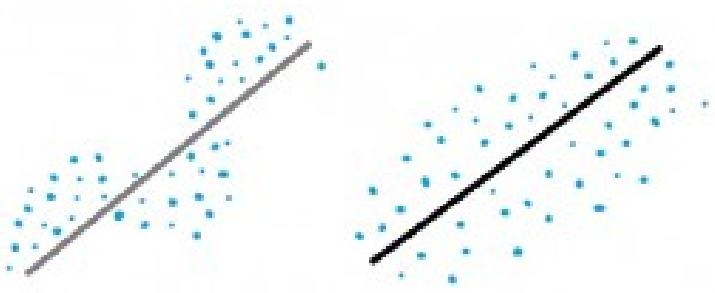
Slika 10. Ravnomerna raspodela grešaka u zavisnosti od veličine projekta.

Sa leve strane slike 10 se vidi raspodela grešaka koja ne prelazi neke maksimalne granice bez obzira na nezavisnu promenljivu. Idealan predikcioni model sa homoskedastičnim greškama je prikazan na desnoj slici.

Treba napomenuti da se testiranje ne može sprovesti vizuelno pošto grafici mogu da prevare, tako da je potrebno primeniti posebne statističke testove kojima se testira homoskedastičnost.

2.3.6.3. Nezavisnost grešaka

Poslednji kriterijum koji rezultati procene moraju da zadovolje je da su greške nezavisne [Kutner *et al.*, 2004]. U dobrim predikcionim modelima greške procene su ravnomerno raspoređene oko idealne predikcione linije, kao što je prikazano na slici 11.



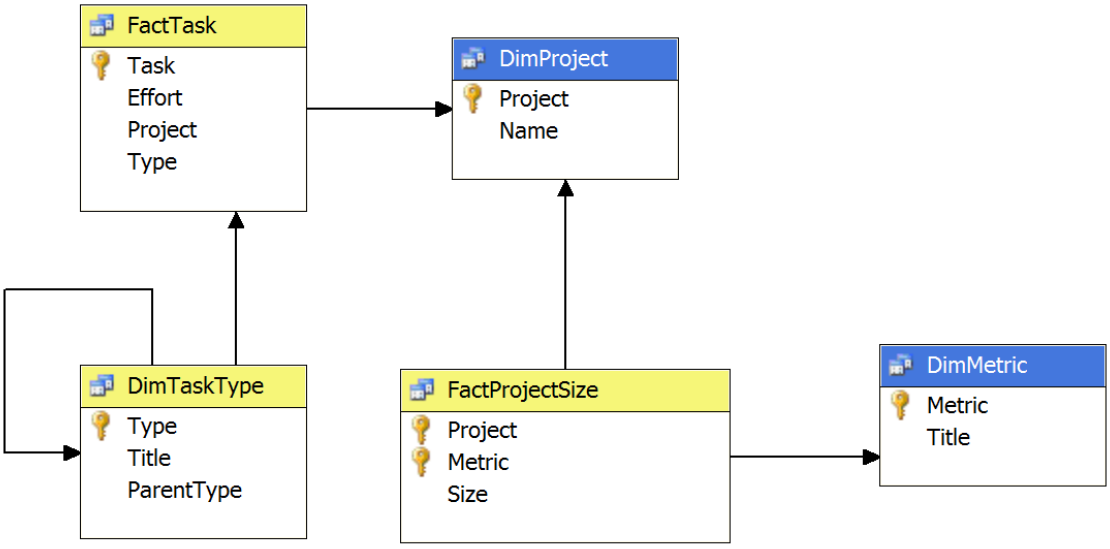
Slika 11. Poređenje grešaka sa i bez međusobne zavisnosti.

Na desnoj strani slike se može primetiti određena zakonitost u raspodeli grešaka što indicira da odabrani model nije idealan za dati skup podataka. Predikcioni model u kome se može primetiti nekakva zakonitost u greškama se mora odbaciti.

2.4. Skladištenje podataka

U istraživanju su korišćeni različiti podaci koji su sačuvani u MS Excel, CSV, ili Google Spreadsheets fajlovima. Kao što će biti opisano u sekciji 2.5., MS Excel i Google Spreadsheets su alati pogodni za analizu i prezentaciju podataka.

Za napredniju analizu kompleksnijih struktura podataka se koristi multi-dimenzionalna struktura [MDX, 2012] prikazana na slici 12.



Slika 12. OLAP multi-dimenzionalni model analizu podataka.

Model OLAP kocke [MDX, 2012] sadrži dve vrste entiteta – činjenice i dimenzije. Činjenice (engl. *Facts*) sadrže podatke koji će se analizirati. Na slici su prikazane dve činjenične tabele FactTask koja sadrži informacije o projektnim zadacima i FactProjectSize koja sadrži informacije o veličinama projekata. Činjenice sadrže informacije koje se agregiraju i prikazuju po različitim kriterijumima. U primeru na slici 12 činjenica FactTask tabela ima informacije o naporu uloženom po svakim projektnom zadatku. Informacije o uloženim naporima po različitim projektnim zadacima se sabiraju i agregiraju kako bi se nad njima vršila analiza.

Dimenzije (engl. *Dimensions*) sadrže podatke po kojima će se vršiti analize. U ovom slučaju to su projekti, vrste projektnih zadataka i metrike kojima su kvantifikovani projekti. One služe da se informacije u činjenicama filtriraju po njima. Na primer, pomoću OLAP model prikazanog na slici 12, se može odrediti suma svih napora u projektnim zadacima filtrirana po dimenziji projekata predstavlja ukupan napor uložen na projektu. Suma svih napora po tipu projektnih zadataka predstavlja ukupno vreme potrošeno na analizu, razvoj testiranje i slično.

U multi-dimenzionalnim kockama se uspostavljaju relacije među dimenzijama i činjenicama kako bi se znalo po kojim kriterijumima (dimenzijama) se mogu agregirati podaci u činjenicama. Za uspostavljanje relacije se koriste identifikatori podataka u činjenicama i dimenzijama koji su slični primarnim i stranim ključevima u modelima relacionih baza. U primeru na slici 12 su prikazane relacije među činjenicama i dimenzijama po kojima se mogu agregirati podaci.

2.5. Alati korišćeni za obradu podataka

U ovoj sekciji će biti predstavljeni alati koji su korišćeni tokom analize rezultata.

2.5.1. Alati za obradu podataka Microsoft Excel i Google Spreadsheets

Microsoft Excel [MSOffice] i Google Spreadsheets [GSHelp] alati za obradu podataka imaju ugrađenu podršku za određivanje statističkih parametara skupova podataka i vizuelizaciju rezultata. Oni omogućavaju kreiranje regresionih modela definisanjem linija trenda (engl. *trendline*) nad podacima. U ovim alatima se lako mogu odabrati različiti oblici regresije, definisati neki parametri modela i vizuelno posmatrati rezultati.

Ovi alati imaju i veliki broj ugrađenih funkcija za analizu podataka koji se mogu koristiti radi procene greške i slično. Koeficijenti linearne regresije se mogu dobiti primenom funkcija koje su ugrađene u ove aplikacije kao što je prikazano u formulama (20) i (21).

$$k = SLOPE([y_n, y_1],[x_n, x_1]) \tag{20}$$

$$n = INTERCEPT([y_n, y_1],[x_n, x_1]) \tag{21}$$

U ovim funkcijama se na osnovu niza podataka nezavisnih promenljivih [x_n, x_1] i odgovarajućeg niza podataka zavisnih promenljivih [y_n, y_1] određuju koeficijenti linearne

regresije. Na ovaj način je izuzetno jednostavno izvršiti analizu pod uslovom da su prikupljeni podaci o projektima i pronaći odgovarajuću funkcionalnu zavisnost.

Ovi alati obezbeđuju veliki broj različitih vrsta grafika za prezentovanje rezultata, tako da predstavljaju dobar izbor za prezentaciju rezultata.

2.5.2. R jezik za statističku analizu

R jezik je popularan jezik za naprednu statističku analizu podataka [R, 2004]. Osnovna struktura podataka u R jeziku je okvir podataka (engl. *dataframe*) koji sadrži matricu podataka u kojoj su sve kolone imenovane. Podaci u okviru se lako mogu učitati iz različitih fajlova ili sistema (npr. baza podataka), kao što je prikazano u sledećem listingu:

```
setwd("C:\Podaci\Analiza\")
ds= read.csv("Podaci.csv")
head(ds)

ds$Test<-ds$FTest+ds$NFTest
```

U primeru se postavlja radni direktorijum, pa se iz njega učitava okvir podataka iz fajla „Podaci.csv“. Funkcija *head()* prikazuje zaglavlje okvira sa kolonama gde se mogu videti imena kolone po kojima se mogu referencirati podaci (npr. kolone sa nazivima FTest i NFTest se referenciraju sa ds$FTest i ds$NFTest). Operatorima dodele se mogu sabirati podaci u kolonama i smeštati rezultati u nove ili postojeće kolone.

U obilju ugrađenih funkcija koje nudi su i funkcije za kreiranje modela na osnovu linearne regresije prikazane u sledećem listingu:

```
model1 <- lm(y ~ x, ds)
model2 <- lm(y ~ x1+x2+x3+x4, ds)
model3 <- lm(y ~ x+x*x+x*x*x, ds)
```

Prva naredba kreira linearni model kojim se predviđaju vrednosti kolone y u zavisnosti od kolone x. Kolone x i y se nalaze u okviru pod nazivom ds.

U drugoj naredbi se kreira linearni model na osnovu skupa nezavisnih promenljivih. Promenljiva y koja se nalazi sa leve strane znaka ~ predstavlja vrednost koja se

procenjuje, dok je skup nezavisnih promenljivih sa desne strane odvojen znakovima +. Kao i u prethodnom slučaju sve promenljive su kolone u skupu ds koji se predaje funkciji.

U trećoj naredbi se kreira nelinearni (polinomski) model. R nema posebne funkcije za kreiranje nelinearnih modela – nelinearni modeli se kreiraju tako što se kao skup nezavisnih promenljivih predaju transformacije nezavisne promenljive i funkcija lm će pronaći optimalne konstante.

Procena vrednosti zavisnih promenljivih na osnovu neke nezavisne promenljive se vrši funkcijom *predict()* kojoj se prosleđuju linearni model na osnovu kojeg se vrši procena kao i vrednost na osnovu koje se daje procena. Pod pretpostavkom da je napravljen model koji na osnovu jedne vrednosti vrši procenu, može se koristiti sledeća naredba:

```
procena <-predict(model, data.frame(x=90))
```

Funkcija *predict()* očekuje okvir podataka kao ulazni parametar tako da se pomoću funkcije *data.frame()* može napraviti okvir sa jednom kolonom sa nazivom x kojoj će se postaviti jedna vrednost na osnovu koje se vrši procena.

Pored funkcija za kreiranje linearnih modela, R jezik ima i funkcije za kreiranje i treniranje neuralnih mreža. Primer funkcije koja kreira i trenira neuralnu mrežu na osnovu matrice ulaznih promenljivih i matrice očekivanih vrednosti je prikazana u sledećem primeru:

```
nn.model <-mlp (input.set, target.set)
procena <- predict(nn.model, data.frame(x=90, y=100, z=120))
```

U ovom primeru se kreira neuralna mreža sa više nivoa (engl. *Multilayer perceptron network*) [Rosenblatt, 1958] pomoću *mlp()* funkcije iz RNSSN paketa [Bergmeir and Benítez, 2012]. Funkcija *predict()* vraća procenjenu vrednost na osnovu treniranog modela i vrednosti (90,100,120).

R jezik ima i veliki broj funkcija kojima se vrši evaluacija predikcionih modela. Da bi se proverilo da li je regresivni model validan, potrebno je da se testiraju sledeće hipoteze [R, 2004], [Fox and Weisberg, 2011]:

1. H_{01}: Greške su normalno distribuirane, što se testira Shapiro-Wilk testom,
2. H_{02}: Greške su homoskedastične, što se testira Breush-Pagan testom,
3. H_{03}: Greške moraju da budu nezavisne, što se testira Durbin-Watson testom.

Svaki regresivni model koji ne prođe testove se odbacuje i zaključuje se da se napor ne može proceniti primenom linearne regresija na osnovu veličine projekta.

Normalna raspodela grešaka se može proveriti pomoću *Shapiro-Wilks* testa koji se primenjuje na nizu rezidualnih grešaka. Funkcija *residuals()* vraća greške procene kreiranih modela i one se mogu proslediti *shapiro.test()* funkciji:

```
shapiro.test(residuals(model))
```

Breuš-Pagan test iz paketa CAR se može koristiti radi testiranja homoskedastičnosti (konstantne varijanse grešaka) [Fox and Weisberg, 2011]. Nulta hipoteza da model nema konstantnu varijansu se testira pomoću sledećeg skripta:

```
library(car)
ncvTest(model)
```

U slučaju da je p vrednost koju je vratila funkcija veća od 0.05, odbacuje se nulta hipoteza da varijacija grešaka nije konstantna.

Nezavisnost grešaka se može proveriti *Durbin-Watson* testom [Fox and Weisberg, 2011]. U paketu CAR se nalazi *durbinWatsonTest()* funkcija kojoj se prosleđuje objekat koji predstavlja predikcioni model, kao što je prikazano u sledećem listingu:

```
durbinWatsonTest(model)
```

Posmatra se p vrednost (engl. *p-value*) vraćenog objekta koji predstavlja regresioni model. U slučaju da je p vrednost veća od 0.05 ne odbacuje se hipoteza da greške nisu normalno raspoređene i prihvata se model.

2.5.3. SQL Server Analitički Servisi (SSAS)

Microsoft SQL Server Analitički Servisi (SSAS) [MDX, 2012] pružaju podršku za naprednu analizu podataka hijerarhijski organizovanih i podeljenih u dve klase entiteta:

1. Činjenice (engl. *Facts*) koje sadrže podatke koji će se analizirati. U ovom istraživanju to mogu biti utrošena vremena na postojećim projektima.
2. Dimenzije (engl. *Dimensions*) koje sadrže podatke po kojima će se vršiti analize u ovom slučaju to su veličine projekata, uticaj nefunkcionalnih zahteva i slično.

Ovakva struktura je optimizovana za relacione strukture podataka koje se ne mogu predstaviti kao jednostavne matrice podataka. Pored struktura za skladištenje podataka, SSAS obezbeđuje i poseban jezik (MDX – *MultiDimensional eXpressions*) za pisanje upita i analizu podataka koji ima ugrađene funkcije za kreiranje modela linearne regresije [MDX, 2012]. Primer upita koji procenjuje napor na projektu linearnom regresijom je prikazan u sledećem upitu:

```
WITH
MEMBER Estimate AS
        LinRegPoint([Project Size], Projects, Effort, [Project Size])
MEMBER R2 AS
        LinRegR2(NONEMPTY(Projects, Effort), [Project Size], Effort)
SELECT        {[Project Name], [Project Size], Estimate, R2} ON ROWS
FROM MeasurementCube
```

Slika 13. MDX upit za procenu napora na projektima.

U ovom primeru je napravljena procena na osnovu *LinRegPoint* funkcije koja na osnovu nezavisnih promenljivih [Project Size] iz skupa projekata *Projects*, predviđa vrednosti napora. Kao ulazni podatak se koriste veličine projekata [Project Size]. Pored samog modela, MDX ima i ugrađenu funkciju za određivanje R^2 koeficijenta. MDX je upitni jezik koji kao rezultat vraća skup podataka slično standardnom SQL jeziku.

SSAS nema prezentacione mogućnosti pošto je njegova osnovna namena skladištenje i analiza podataka. Međutim bilo koji prezentacioni alat kao što je Excel se jednostavno može povezati sa SSAS sistemom i preuzeti podatke (tj. rezultate upita) koje treba prikazati.

2.6. Eksperimentalni podaci

U analizi je korišćeni realni podaci o komercijalnim projektima implementirani u softverskoj kompaniji koja se bavi razvojem sistema za klijente u Velikoj Britaniji, Holandiji i Nemačkoj. Analiza je izvršena nad 50 ASP.NET aplikacija implementiranih u C#/ASP.NET tehnologiji. Statistike projekata su prikazane u tabeli 3.

Tabela 3. Statistički parametri o skupu projekata korišćenih u radu.

	Napor (Čovek*meseci)	Trajanje (Meseci)	Veličina tima (Zaposlenih)	Iskustvo (Godina)
Minimum	2	2	4	1
Prvi kvartil	4	4	6	4
Medijana	7	5	7	5
Treći kvartil	12	7	8	6
Maksimum	15	12	10	8

Kao što se može primetiti korišćen je skup srednjih i malih projekata (srednja vrednost je 125 čovek-dana) od 40 do 400 čovek-dana. Polovina projekata je u opsegu od 57 do 178 čovek-dana. Iako su nad ovim projektima korišćeni različiti modeli razvoja kako formalni RUP (24 projekata) [Kruchten, 2000], MSF(8 projekata) [MSF, 2003], [MSF, 2008] tako i iterativni/agilni (17 projekata) metodologija nije uticala na vreme razvoja. U istraživanju u radu su korišćeni sledeći dokumenti kao izvori informacija:

1. Dokument vizije – u projektima koji su analizirani projektni tim je imao obavezu da na početku projekta napiše viziju projekta u kojoj se nalazi okvirni opis funkcionalnosti i modela podataka.
2. Dokument procene vremena – deo projektnog plana je ili poseban dokument ili sekcija u kojoj se nalazi vreme koje je planirano za projekat.
3. Korisnička dokumentacija, uputstva i prototipovi – dokument sa informacijama o izgledu strana i funkcionalnostima se koristi radi procene kompleksnosti.
4. Modeli baze podataka – logički modeli baza podataka dizajniranih u projektima su deo tehničke specifikacije. Modeli predstavljaju izvor informacija o kompleksnosti podataka u sistemu.
5. Izveštaj o radu – za projekte koji su rađeni iterativno postoje izveštaji o vremenu potrošenom za implementaciju funkcionalnosti u okviru iteracija.
6. Izveštaji iz alata kojima se prate aktivnosti na projektu.

3. Pregled metoda procene veličine softvera

U prvom delu istraživanja će biti evaluirane metode kojima se kvantifikuje veličina softverskih sistema sa stanovišta mogućnosti njihove primene radi procene napora. S obzirom da su metode ekspertske i timske procene napora trenutno dominantne u praksi kada se procenjuje napor koji je potreban da se implementira sistem, potrebno je ispitati da li postoje i druge metode koje mogu pomoći u procesu procene i koliko su te metode primenljive. Trenutno postoji nekoliko metoda kojima se može oceniti veličina softvera koje se mogu podeliti u sledeće grupe:

1. Metode kojima se procenjuje obim posla analizom količine programskog koda koji je potrebno napisati kako bi se implementirao sistem,
2. Metode kojima se procenjuje obim posla procenom količine funkcionalnosti koja bi trebalo da bude implementirana u sistemu.

U ovom poglavlju će biti prikazane metode za analizu i kvantifikaciju koda i funkcionalnosti. Pored toga, najčešće korišćene metode će biti upoređene sa stanovišta preciznosti i greške procene napora.

3.1. Metode zasnovane na analizi koda

Metode merenja veličine sistema analizom koda se fokusiraju na sam programski kod koji se kreira tokom izrade sistema. Ovo su bile prve mere koje su korišćene za merenje veličine softvera. Tri najvažnije metode iz ove grupe su merenje broja linija koda [Park, 1992], ciklomatske kompleksnosti [McCabe, 1976] i Halstedova mera [Halstead, 1997].

3.1.1. Brojanje linija koda

Prva mera koja se koristila radi merenja veličine softvera je bila broj linija koda [Park, 1992]. Ideja za ovu meru je došla na osnovu analogije sa ostalim merama za merenje veličine dokumenata kao što je broj strana dokumentacije, broj reči u tekstu i slično. Broj linija koda je bila veoma pogodna i merodavna mera u proceduralnim programskim jezicima gde je broj linija koda mogao manje-više verno da odslika vreme koje je potrebno da se programski kod unese, naravno pod pretpostavkom da prate postojeći algoritmi ili da je vreme za osmišljavanje algoritama srazmerno količini programskog koda. Najznačajnija metoda za procenu napora na osnovu veličine softvera COCOMO II

[Boehm *et al.*, 2000] koristi broj linija koda kao jedinicu mere veličine softvera na osnovu koje se procenjuje vreme potrebno da se implementira sistem.

Linije koda pored svoje jednostavnosti i lakoće određivanja imaju niz mana. Jedna od mana je utvrđivanje pravila definicije linije koda. Kao primer određivanja broja linija na različit način u zavisnosti od formata koda može se koristiti kod na slici 14.

while(i<10) printf(i);	int i;
	for(i=0; i<10; i++)
	{
	// Štampanje broja
	printf(i);
	}

Slika 14. Ekvivalentan kod sa različitim brojem linija koda.

Iako je programski kod identičan, na desnoj strani bi se alatom za određivanje veličine koda dobila šest puta veća veličina.

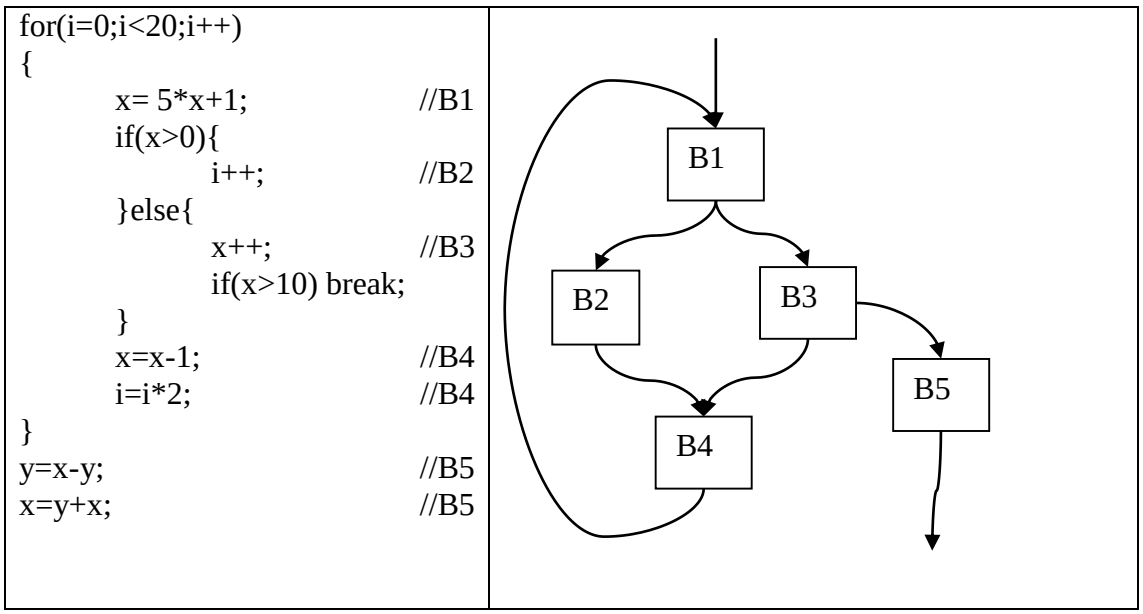
Linije koda se teže mogu primeniti u heterogenim sistemima softvera gde se kombinuje kod napisan u više programskih jezika (C++, Java, C#). Pored toga, u jezicima koji se razliku po nameni kao što su JavaSkript i SQL nije moguće uspostaviti nikakvu ekvivalenciju. JavaSkript nema mogućnost čitanja podataka iz baza podataka, dok SQL nema mogućnost prezentacije rezultata tako da u ovakvim slučajevima nije moguće napraviti nikakvu ekvivalenciju po pitanju odnosa veličine i linija koda.

U sistemima koji koriste već gotove biblioteke funkcija se mogu relativno malim brojem linija koda implementirati veoma složene funkcionalnosti s obzirom da se u kodu vrši samo pozivanje gotovih funkcija. U programskim jezicima koje nemaju gotove biblioteke za istu funkcionalnost mora implementirati deo paketa koji već postoji u drugim jezicima.

Imajući u vidu nedostatke broja linija koda može se primetiti da to nije pogodna mera za određivanje veličine softvera, sem u nekim specifičnim slučajevima. Broj linija koda se može koristiti u homogenom softveru koji nije implementiran pomoću gotovih biblioteka i više različitih programskih jezika – primer takvog softvera su operativni sistemi. Sem u merenju veličine operativnih sistema i sličnog sistemskog softvera, mera broja linija izvornog koda nije naišla na veću primenu u ostalim tipovima softverskih aplikacija.

3.1.2. Ciklomatska kompleksnost

Alternativa direktnoj analizi linija koda je predstavljanje koda ekvivalentnim modelima pomoću kojih se vrši analiza kompleksnosti. Programski kod se može predstaviti grafovima toka kontrole [Aho *et al.*, 1986], [Allen, 1970] koji su nezavisni od konkretnog programa i mogu se efikasnije koristiti za analizu kompleksnosti sistema. Primer grafa toka kontrole za deo programskog koda je prikazan na slici 15.



Slika 15. Primer grafa toka kontrole programskog koda.

Blokovi na slici 15 predstavljaju sekvence neprekidnih naredbi u izvornom kodu. Iz jedne sekvence u drugu se može preći nekom naredbom skoka, grananjem u uslovnim naredbama, petljama ili bilo kojom naredbom kojom se prelazi u neku drugu liniju koda. Detaljna uputstva za formiranje grafova toka kontrole na osnovu izvornog koda su data u [Aho *et al.*, 1986].

Pored standardnih grafova toka kontrole koji se koriste u standardnim proceduralnim jezicima [Allen, 1974] tokovi kontrole se mogu napraviti za objektno orijentisani kod [Berg *et al.*, 1999], [Harrold and Rothermel, 1994], distribuirane aplikacije [Dwyer, 1997], aspektno-orijentisane sisteme [Zhao *et al.*, 2006], itd. Pošto nema ograničenja koja bi mogla uticati na primenu predstavljanja koda grafovima toka kontrole svaki programski kod se može predstaviti ovakvom formom.

Tomas MekKejb [McCabe, 1976] je definisao meru ciklomatske kompleksnosti izvornog koda koja se određuje na osnovu parametara ekvivalentnog grafa toka kontrole. Ciklomatska kompleksnost meri broj linearno nezavisnih putanja u grafu toka kontrole pomoću formule (22).

$$M = E - N + 2P \tag{22}$$

U ovoj formuli M je ciklomatska kompleksnost koda, E broj ivica u grafu toka kontrole, N broj čvorova, dok je P broj povezanih komponenti.

Iako je u nekim radovima [Stain *et al.*, 2005] pokazano da postoji jaka korelacija između ciklomatske kompleksnosti i ekspertske procene ova metoda nije naišla na šire prihvatanje u softverskoj industriji.

Ciklomatska kompleksnost se primenjuje radi procene napora i vremena potrebnog za testiranje [Watson and McCabe, 1996]. Tokom testiranja softvera se veoma često kao polazna tačka uzimaju grafovi tokova kontrole ili grafovi toka podataka kako bi se identifikovali testovi. S obzirom da su u ovom procesu na raspolaganju grafovi toka kontrole, može se koristiti ciklomatska kompleksnost radi procenu obima posla tokom testiranja. Pored toga, ova mera se često koristi u nekim integrisanim razvojnim okruženjima prilikom analize programskog koda u cilju otkrivanja previše kompleksnih programskih celina.

3.1.3. Halstedova metrika

Mauris Halsted je 1977. godine predstavio svoju metriku kao još jednu alternativu brojanju linija koda [Halstead, 1997], [Zuse, 2005]. Halstedova metrika se može primeniti direktno na programski kod tako što se analiziraju određeni elementi u njemu. Kod se apstrahuje skupom operanada koji predstavljaju objekte u kodu (npr. promenljive, parametri, fajlovi, slike) i operacija koje se vrše nad operandima (npr. aritmetičke operacije, funkcije, ili bilo koji drugi procesi koji rade sa operandima). Kada se programski kod zameni skupom ekvivalentnih operanada i operacija određuje se ukupan broj operatora u sistemu (n1), ukupan broj operanada (n2), broj različitih operatora u sistemu (N1) i broj različitih operanada (N2). Veličina softvera (*Volume* – V) i složenost (*Difficulty* – D) se određuju pomoću ovih parametara prema formulama (23) i (24).

$$V = (N_1 + N_2) * \log_2(n_1 + n_2) \quad (23)$$

$$D = \frac{n_1}{2} * \frac{N_2}{n_2} \quad (24)$$

Ukupan napor koji je potrebno uložiti na izradu sistema je srazmeran proizvodu veličine i složenosti. Halsted [Halstead, 1997] je ovu veličinu predstavio kao metriku koja verno može da kvantifikuje sistem, međutim neka istraživanja pokazuju da ova metrika nije dobro korelisana sa naporom potrebnim za implementaciju projekta [Lind and Vairavan, 1989]. Halstedova metrika, kao ni ciklomatska kompleksnost nije šire prihvaćena u softverskoj industriji.

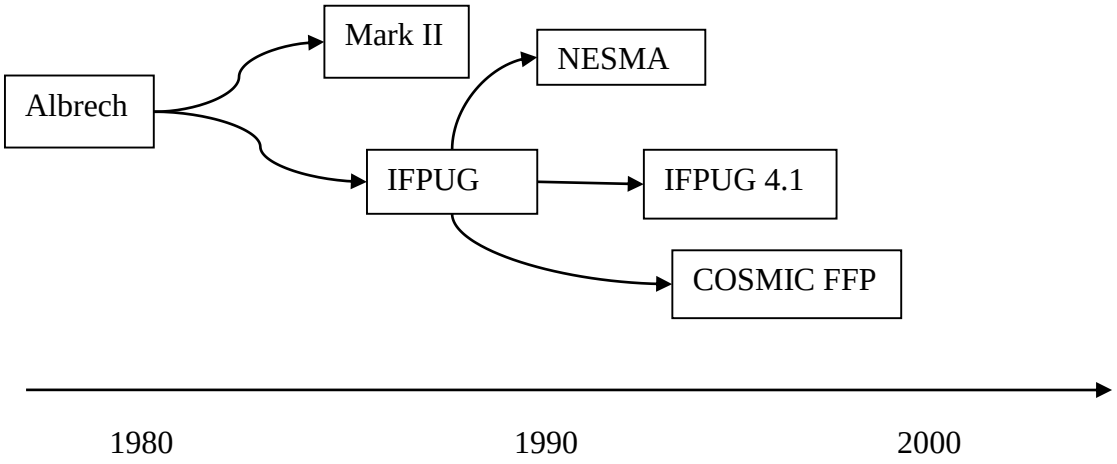
Metode zasnovane na merenju funkcionalnosti korišćenjem funkcionalnih tačaka koje su kasnije razvijene su preuzele primat nad metodama analize koda, tako da se Halstedova metrika kao i ostale metode iz ove familije koriste uglavnom u integrisanim okruženjima kao pomoćno sredstvo u analizi koda i otkrivanju kompleksnih modula što više ima veze sa kontrolom kvaliteta koda nego sa merenjem veličine softvera. Zbog toga u praksi ne postoji veći broj podataka o uspešnosti primene ove metode.

3.2. Metode zasnovane na analizi funkcionalnosti

Kako bi se prevazišao problem merenja sistema analizom koda, istraživači su pokušavali da nađu ekvivalentnu meru koja bi ne bi imala probleme tehnološke prirode koje imaju linije koda. Prva ideja je bila da se umesto programskog koda analiziraju funkcionalnosti sistema čime je stvorena cela familija metoda baziranih na novoj meri – funkcionalnim tačkama [Popovic, 2010]. Ove metode teže da odrede veličinu koja kvantifikuje količinu funkcionalnosti u sistemu (tj. funkcionalnu veličinu koja će u nastavku biti označena sa *F-Size*).

Jedan primer funkcionalne veličine su funkcionalne tačke. Funkcionalne tačke su se pokazale kao efikasan način za predstavljanje veličine projekta pošto je jedinica mere koja se koristi bazirana na funkcionalnostima a ne na tehnologiji. Različiti sistemi koji implementiraju slične funkcionalnosti koristeći različite tehnologije ili programske jezike se mogu razlikovati u broju linija koda koje su potrebne za implementaciju, ali će zato imati iste veličine izražene u funkcionalnim tačkama. Sa korisničke strane, ovo je

značajnija mera, pošto se vrednost sistema ocenjuje na osnovu funkcionalnosti koje daje korisnicima a ne na osnovu tehnologije. Najbitnije metode iz ove grupe su prikazane na slici 16 [Popovic, 2010].



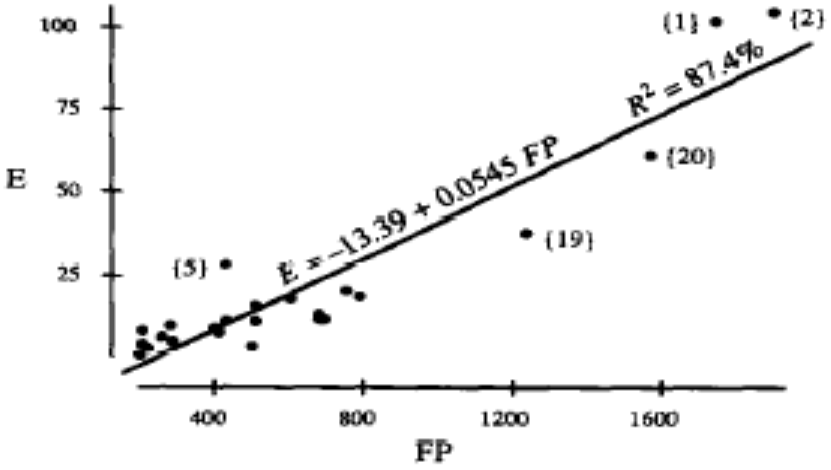
Slika 16. Evolucija funkcionalnih metoda merenja veličine softvera.

Prve pokušaje definisanja mere koja je nezavisna od tehnologije i bazira se na merenju veličine na osnovu funkcionalnosti su sproveli Albreht, Gafni [Albrecht and Gaffney, 1983] i Kemerer [Kemerer, 1987], [Kemerer and Porter, 1992] krajem osamdesetih godina dvadesetog veka. Albreht je pokušao da merenjem funkcionalnosti u IBM projektima utvrdi funkcionalnu zavisnost između veličine projekta i vremena potrebnog za izradu. Radi merenja veličine projekta definisao je meru veličine funkcionalnosti koju je nazvao funkcionalna tačka (engl. *Function point* – FP). Ovaj princip kvantifikovanja funkcionalnosti sistema je prihvaćen i od strane ostalih istraživača.

Albrehtova metoda je kasnije dalje razvijana od strane IFPUG-a (*International Function Point User Group*) [IFPUG, 1994] koji danas predstavlja osnovni pravac daljeg razvoja funkcionalnih tačaka i NESMA grupe (*NEtherland Software Measurement Group*) [NESMA, 1997] koja je napravila veoma sličnu metodologiju merenja sistema tehnikom funkcionalnih tačaka. U Velikoj Britaniji se koristi varijanta FP metode koja se naziva Mark II [Symons, 1991], [UKSMA, 1998]. Najnovija metoda koja bi trebala da prevaziđe nedostatke standardne IFPUG metode je COSMIC FFP metoda [Abran *et al.*, 1999], [Symons and Rule, 1999], [Abran *et al.*, 2004]. U ovoj sekciji su opisane najvažnije metode merenja iz familije funkcionalnih tačaka.

3.2.1. Funkcionalne tačke (Albreht/Gafni i Kemerer)

Albreht i Gafni [Albrecht and Gaffney, 1983], [Albrecht, 1994] su sakupili podatke o vremenima implementacije 24 projekta implementirana u IBM-u i pokušali da pronađu zavisnost između vremena izrade i funkcionalne veličine. Na osnovu funkcionalnih veličina i trajanja projekata vršili su analizu zavisnosti između ovih veličina korišćenjem linearne regresije. Slika 17 [Albrecht and Gaffney, 1983] prikazuje rezultate koje su dobili linearnom regresijom nad prikupljenim podacima.

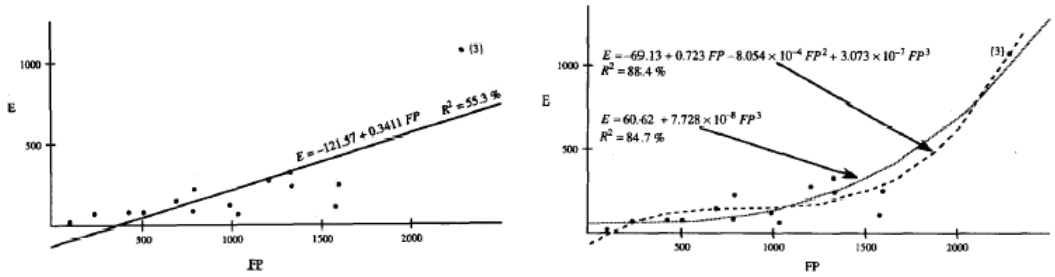


Slika 17. Linearna zavisnost između funkcionalne veličine i napora.

Funkcionalna zavisnost koju su dobili je relativno pouzdana (R^2 = 87.40%), ali ipak nije bilo moguće potvrditi rezultate s obzirom na mali broj parova podataka koji su analizirani.

Bez obzira na moguću nepouzdanost rezultata, rad Albrehta i Gafnija je bio početak primene metode funkcionalnih tačaka i značajan korak u metodama analize zavisnosti među parametrima sistema.

Kemerer [Kemerer, 1987] je na osnovu ove mere i podataka iz 15 projekata definisao nelinearni model zavisnosti veličine softvera i vremena potrebnog za implementaciju. Rezultati Kemererove analize su prikazani na slici 18. Pouzdanost modela pri linearnoj regresiji je bila 55.3%, dok u slučaju korišćenja regresivnog modela trećeg stepena pouzdanost prelazi 85%.



Slika 18. Kemererovo poboljšanje predikcionog modela nelinearnom regresijom.

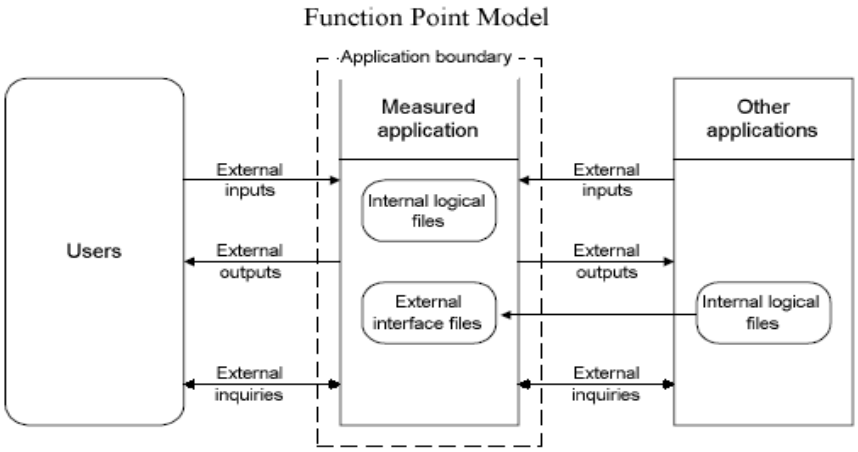
Radovi Albrehta i Kemerera su predstavljali polaznu tačku u definisanju nove mere softverskih sistema koja će prevazići nedostatke merenja sistema linijama koda. IFPUG [IFPUG, 1994], NESMA [NESMA, 1997] i druge grupe su razvile nove metode i usavršile pravila tako što su analizirali mnogo veći skup projekata sakupljenih od raznih softverskih kompanija u svetu i tako napravile dosta pouzdanije modele.

3.2.2. IFPUG metoda (FPA)

IFPUG je internacionalna grupa koja se sastoji od više grupa podeljenih po zemljama kao što su Velika Britanija, Brazil, Australija i tako dalje. IFPUG metodologija merenja veličine softvera funkcionalnim tačkama [IFPUG, 1994] predstavlja dalje razvijanje ideja koje je koristio Albreht.

IFPUG je globalna organizacija koja koristi veliki broj projekata u svojoj analizi. Detaljna pravila za definisanje veličine sistema su definisana u priručniku za prebrojavanje funkcionalnih tačaka [IFPUG, 1994] i ISO standardu [ISOFSM, 2007]. U ovoj sekciji opisan pregled FPA pravila.

Prilikom merenja veličine sistema metodom funkcionalnih tačaka u sistemu se sakupljaju informacije o podacima (fajlovima) i procesima (transakcijama) koje su prikazane na slici 19 [IFPUG, 1994].



Slika 19. Model merenja veličine sistema FPA metodom.

U FPA metodi se identifikuju logičke strukture podataka koje sadrže informacije sa kojima sistem radi i procenjuje se njihova složenost. Logičke strukture podataka (koje se nazivaju fajlovi u FPA metodi) mogu biti interni logički fajlovi koji predstavljaju podatke koji se nalaze u sistemu koje aplikacija održava ili eksterni logički fajlovi koji predstavljaju podatke koji se nalaze u drugim sistemima i koje komponente sistema koriste pri radu. Pošto se identifikuju fajlovi potrebno im je oceniti veličinu na osnovu dve osnovne karakteristike:

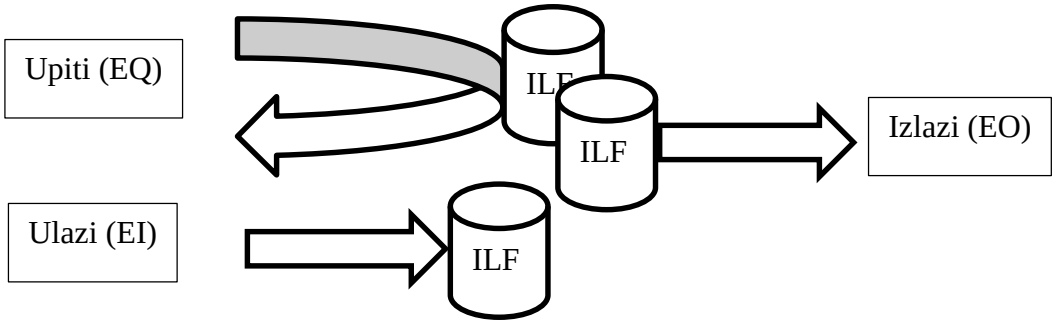
1. Broj podataka u fajlu (engl. *Data Element Types* – DET) koji predstavlja broj jednostavnih logičkih struktura podataka koje predstavljaju logički nedeljive – atomske podatke.
2. Broj zapisa (engl. *Record Element Types* – RET) koji predstavlja interne grupe informacija u okviru jednog fajla koje su ili logički povezane ili se mogu ponavljati.

Kombinacijom broja podataka i zapisa fajlovi se kategorizuju fajlovi kao fajlovi niske, srednje ili visoke kompleksnosti na osnovu tabele 4 [IFPUG, 1994].

Tabela 4. Određivanje kompleksnosti fajlova na osnovu broja podataka i zapisa.

	DET		
RET	1-19	20-50	>50
1	Niska	Niska	Srednja
2-5	Niska	Srednja	Velika
>5	Srednja	Velika	Velika

Drugi faktor koji utiče na složenost sistema su transakcije kojima se odvija komunikacija među korisnicima i podacima u okviru sistema. Transakcije su funkcionalnosti koje predstavljaju tokove podataka kojima se podaci unose u sistem, kojima se podaci izvlače iz sistema i tokove kojima se za određeni ulaz bez obrade vraćaju podaci (upiti). Tri osnovna tipa transakcija koji se posmatraju u FPA metodi su prikazani na slici 20.



Slika 20. Eksterni ulazi (EI), eksterni izlazi (EO) i upiti (EQ) u FPA metodi.

Generalno, transakcije se mogu podeliti na tri vrste u zavisnosti od načina na koji obrađuju podatke:

1. Ulazni procesi (engl. *External Inputs* – EI) kojima se podaci unose u sistem i interne logičke fajlove (dodavanje novih informacija, ažuriranje ili brisanje postojećih informacija).
2. Izlazni procesi (engl. *External Outputs* – EO) kojima se čitaju podaci iz internih ili eksternih fajlova, obrađuju i prenose van sistema. Osnovna karakteristika izlaznih procesa je činjenica da se nad podacima koji se šalju van sistema mora vršiti nekakva obrada. Oni često imaju i ulazne parametre koji mogu definisati izlazne podatke.
3. Upiti (engl. *External Enquiries* – EQ) kojima za određeni skup ulaznih podataka daju rezultati na izlazu bez dodatne obrade. Upiti imaju i ulazne i izlazne tokove podataka kojima se definišu parametri upita i sami podaci u rezultatima.

Kao i fajlovi, transakcije se klasifikuju kao transakcije niske, srednje i visoke složenosti na osnovu sledećih parametara:

1. Broja podataka u transakciji (engl. *Data Element Types* – DET) koji predstavljaju logičke informacije koje se unose u sistem ili iznose iz sistema. U zavisnosti od tipa transakcije (ulazna, izlazna ili upit) ovi podaci se mogu upisivati u fajlove sistema, prikazivati korisnicima, slati drugim sistemima i slično.

2. Broja referenciranih fajlova (engl. *File Type Referenced* – FTR) iz kojih transakcija čita ili upisuje podatke. Skup internih i eksternih fajlova koje transakcija koristi predstavlja meru njene složenosti.

Kao i za fajlove, postoje pravila kojima se klasifikuju transakcije prema ovim karakteristikama. Pravila za klasifikaciju eksternih ulaza su prikazana u tabeli 5. Pravila za ostale tipove su slična uz druge granice za referencirane fajlove i podatke [IFPUG, 1994].

Tabela 5. Određivanje kompleksnosti eksternih ulaza.

	DET		
FTR	1-4	5-15	>15
0-1	Niska	Niska	Srednja
2	Niska	Srednja	Velika
>2	Srednja	Velika	Velika

Svakom elementu u FPA modelu se na osnovu procenjene kompleksnosti dodeljuje odgovarajuća težina. U tabeli 6 su prikazane vrednosti težina za sve elemente u modelu u zavisnosti od kompleksnosti.

Tabela 6. Težine koje se dodeljuju elementima FPA modela na osnovu složenosti.

	Jednostavna	Srednja	Složena
Eksterni ulazi	3	4	6
Eksterni izlazi	4	5	7
Eksterni upiti	3	4	6
Interni logički fajlovi	7	10	15
Eksterni logički fajlovi	5	7	10

Sabiranjem veličina svih elemenata u modelu se dobija funkcionalna veličina sistema. Ovakva veličina se u FPA metodi naziva neprilagođena funkcionalna veličina sistema (*Unadjusted Function Point Value* – UFP).

Formalno gledano, veličina sistema se može predstaviti kao skup od pet trodimenzionalnih vektora gde svaki predstavlja broj jednostavnih, srednjih i složenih elemenata za svaki od pet navedenih tipova elemenata. Alternativna predstava je

petnaesto-dimenzionalnim vektorom koji sadrži sve potrebne informacije o složenosti po svim klasama elemenata. U slučaju da se sistem predstavi jednim petnaesto-dimenzionalnim vektorom S, veličina (norma) se određuje formulom (25).

$$UFP = \left\| \vec{S} \right\| = \vec{S} * \vec{W} \tag{25}$$

U formuli (25), težinski vektor W takođe ima petnaest dimenzija i predstavlja skup svih vrednosti iz tabele težina W=(3,4,6, 4,5,7, 3,4,6, 7,10,15, 5,7,10). Na ovaj način je formalno definisana veličina sistema.

FPA metoda je, posle tačaka slučajeva korišćenja [Karner, 1993], druga najčešće korišćena metoda u procesu procene vremena [Kassab *et al.*, 2014]. Veliki broj softverskih kompanija u svetu koristi ovu metodu i ona je postala osnova za merenje veličine softvera. Pored toga ova metoda je osnova za razvoj novih metoda koje se primenjuju u realnim sistemima [St-Pierre *et al.*, 1997] ili održavanju [Niessink, 1997].

3.2.3. NESMA metoda

Početkom devedesetih godina grupa istraživača uz Holandske asocijacije za merenje softvera je napravila modifikacije originalne FPA metode i definisala posebnu NESMA metodu [NESMA, 1997].

Osnovna modifikacija standardne FPA metode je uvođenje tri nivoa pouzdanosti. FPA metoda zahteva detaljnu analizu da bi se odredila veličina sistema. U slučaju da je potrebno dati procene veličine sistema sa ograničenim brojem podataka ili ako nema vremena za detaljnu analizu koju zahteva FPA metoda mora se primeniti neka aproksimativna metoda za procenu veličine sistema. NESMA grupa je u svojoj modifikaciji standardne FPA metode definisala tri različite metode procene:

1. NESMA detaljnu metodu koja je praktično identična FPA metodi. Iako postoje potpuno nezavisni dokumenti koji opisuju ove dve metode u praksi je veoma teško naći značajnije razlike tokom procenjivanja veličine sistema ovim metodama.
2. NESMA procenjenu metodu (NESMA-E) kojom se uzimaju podrazumevane vrednosti za kompleksnosti transakcija i fajlova u sistemu.
3. NESMA indikativna metoda (NESMA-I) kojom se samo na osnovu veličine fajlova procenjuje veličina sistema. Za broj i kompleksnost transakcija se uzimaju podrazumevane vrednosti.

Po NESMA procenjenoj metodi se pretpostavlja da sve transakcije definisane u FPA metodi imaju srednju kompleksnost i da svi interni i eksterni fajlovi imaju malu kompleksnost. Ova pretpostavka omogućava da se samo identifikuju fajlovi i transakcije u sistemu bez ulaženja u detaljnu analizu svake transakcije ili fajla. Na ovaj način sistem se predstavlja vektorom veličine S = (ILF, EIF, EI, EO, EQ) kojim se opisuje samo ukupan broj fajlova i transakcija identifikovanih u sistemu. Veličina sistema se određuje kao norma vektora po formuli (26).

$$UFP = \left\| \vec{S} \right\| = \vec{S} * \vec{W} \tag{26}$$

U formuli (26) se koristi vektor težina W = (7, 5, 4, 5, 4) čime se dobija funkcionalna veličina sistema.

Druga aproksimativna verzija FPA metode je NESMA indikativa metoda (NESMA-I) koja se koristi u veoma ranim fazama projekata. Prebrojavanjem internih i eksternih fajlova bez detaljne analize njihovih karakteristika sistem se opisuje vektorom S = (ILF, EIF). Funkcionalna veličina sistema se dobija kao norma ovog vektora koja se računa na osnovu formule (27).

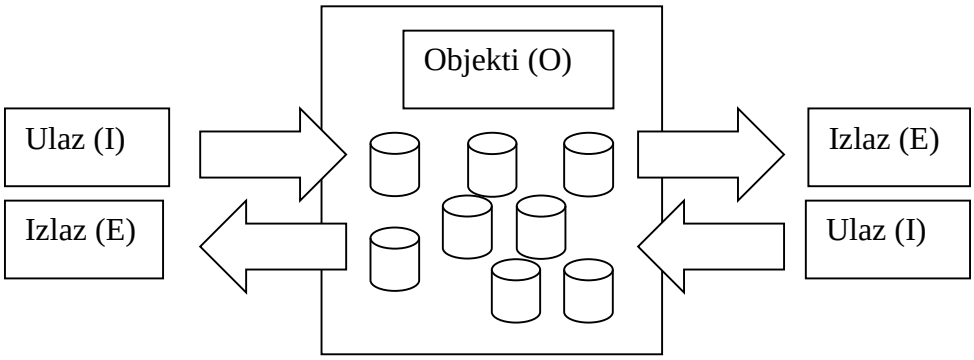
$$UFP = \left\| \vec{S} \right\| = \vec{S} * \vec{W} \tag{27}$$

U formuli (27) se koristi težinski vektor W = (35, 15) pomoću koga se dobija funkcionalna veličina sistema. Osnovna ideja NESMA-I aproksimativne metode leži u pretpostavci da će za svaki interni fajl biti vezana tri ulazna procesa (promena, kreiranje i brisanje podataka), dva izlaza i jedan upit, dok na svaki eksterni fajl u proseku dolazi jedan upit i jedan izlaz.

Iako se razlikuje od standardne IFPUG metode, NESMA metoda je potpuno ravnopravna i takođe sertifikovana kao ISO/IEC 24570:2005 standard. Iako se umesto detaljne NESMA metode češće koristi IFPUG metoda (uglavnom zbog većeg broja podataka na osnovu kojih je metoda napravljena), NESMA procenjena i indikativna metoda predstavljaju nezamenljive metode u ranim fazama projekta kada je potrebno proceniti veličinu sistema na osnovu ograničenog broja podataka.

3.2.4. Mark II

Mark II metoda [Symons, 1991], [UKSMA, 1998] predstavlja modifikaciju originalne Albrehtove metode kojom se umesto pet elemenata sistema posmatraju samo tri elementa – fajlovi, upisi i izlazi. Za razliku od ostalih metoda Mark II metoda ne pravi razliku među internim i eksternim fajlovima i ne poznaje pojam upita (u zavisnosti od toka podataka upit je ili upis ili izlaz). Informacije koje se posmatraju u sistemu su broj fajlova koji se nalaze u okviru sistema, broj procesa koji unose podatke u sistem i broj procesa koji iznose podatke iz sistema kao što je prikazano na slici 21.



Slika 21. Elementi sistema koji se posmatraju u Mark II metodi.

Sistem se opisuje vektorom S = (Ni, Ne, No) koji predstavlja broj ulaza, izlaza i objekata u sistemu, respektivno. Da bi se dobila mera veličine sistema ovaj vektor se skalarno množi sa težinskim vektorom W = (Wi, We, Wo) čime se dobija funkcionalna veličina sistema (F-Size) prema formuli (28).

$$FSize = \left\| \vec{S} \right\| = \vec{S} * \vec{W} \tag{28}$$

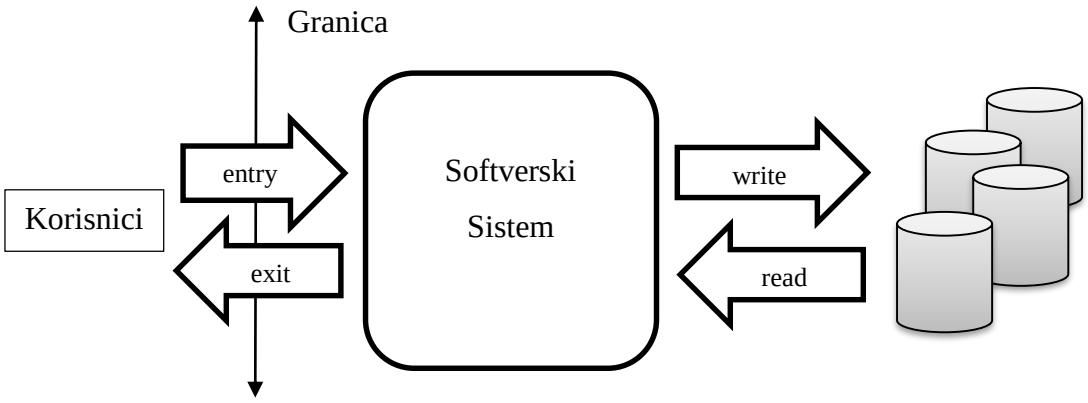
Trenutne preporučene vrednosti težinskih koeficijenata su W = (0.58, 1.66, 0.29). Kao i kod prethodnih metoda ova veličina predstavlja neprilagođenu veličinu u kojoj nije uračunat uticaj nefunkcionalnih faktora.

Mark II metoda po preciznosti odgovara NESMA procenjenoj metodi (NESMA-E) s obzirom da koristi podrazumevane vrednosti za kompleksnosti transakcija i fajlova. Mana Mark II metode je to što ne pravi razlike među internim-eksternim fajlovima i što se upiti moraju poistovetiti sa ulazima ili izlazima. Ovo unosi dodatnu nepreciznost u meru

veličine sistema ali i dalje je moguće koristiti u ranijim fazama projekata. Ova metoda nije toliko prihvaćena kao ostale funkcionalne metode mada u određenim zemljama kao na primer u Velikoj Britaniji ima veliki procenat zastupljenosti. I ova metoda je zvanično sertifikovana kao ISO/IEC 20968:2002 standard.

3.2.5. COSMIC FFP

COSMIC FFP metoda [Abran *et al.*, 1999], [Abran *et al.*, 2004] je najnovija metoda iz familije funkcionalnih metoda kreirana od strane konzorcijuma za merenje softvera (*COmmon Software Measurement International Consortium* – po kome je i dobila ime), u cilju prevazilaženja nekih ozbiljnih problema koji su pogađali postojeće metode iz ove familije. COSMIC FFP metoda posmatra elementarne komunikacione poruke koji se razmenjuju u sistemu [Symons and Rule, 1999]. Procesi koji se posmatraju prilikom analize po COSMIC FFP metodi su prikazani na slici 22.



Slika 22. Elementi koji se posmatraju u COSMIC metodi.

Poruke se mogu razmenjivati između korisnika i sistema, među komponentama sistema, između sistema i skladišta podataka i slično. Postoji četiri osnovna tipa poruka:

1. Ulazi koji predstavljaju poruke koje korisnik šalje u sistem ili koje šalje jedan modul drugom kako bi mu poslao podatke. Ove poruke ne moraju da predstavljaju upise u sistem.
2. Izlazi koji predstavljaju poruke koje sistem ili moduli vraćaju kao odgovor. Podaci mogu da se čitaju iz fajlova, da budu rezultati aritmetičkih operacija, i slično.
3. Upisi koji predstavljaju poruke kojima se ažuriraju podaci u sistemu (tj. fajlovima, tabelama, itd.).

4. Čitanja koji predstavljaju poruke kojima se čitaju podaci iz sistema, tabela i fajlova.

U COSMIC FFP metodi se identifikuju elementarne poruke koje se razmenjuju u sistemu. Funkcionalna veličina sistema je broj poruka koje se koriste. Na ovaj način sistem se predstavlja četvoro-dimenzionalnim vektorom S = (E, X, W, R) koji predstavlja ukupan broj poruka kojima podaci ulaze, izlaze u sistem, upisuju se ili se čitaju iz fajlova. Norma sistema se određuju formulom (29).

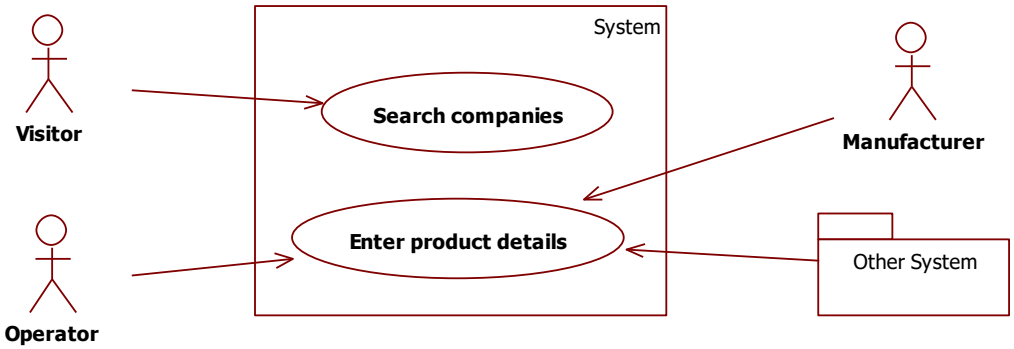
$$\left\|\vec{S}\right\| = E + X + W + R \tag{29}$$

Kao što se vidi u ovoj metodi nema težinskih koeficijenata – primenjuje se obična Menhetn norma opisana u sekciji 2.1.3. Ova metoda se može primenjivati i na monolitne sisteme i na distribuirane i modularne sisteme tako što se u modularnim sistemima broje i poruke koje moduli šalju jedni drugima. I najmanja promena se može detektovati i odrediti njen uticaj na veličinu tako da ne postoji gornja granica veličine funkcionalnosti a samim tim ni zasićenje pošto kompleksnost funkcionalnosti može neograničeno rasti u zavisnosti od broja poruka u sistemu.

COSMIC FFP metoda je najmlađa u familiji funkcionalnih metoda tako da joj je mana manjak podataka o realnim projektima. Međutim očekuje se da se će ona u budućnosti sve više uzimati primat u odnosu na ostale metode ili će se ostale metode menjati tako da se više prilagode COSMIC FFP metodi. COSMIC FFP metoda je standardizovana po ISO/IEC 19761:2003 standardu.

3.2.6. Tačke slučajeva korišćenja

Razvoj UML metodologije [Booch *et al.*, 1998] u savremenim softverskim projektima je naveo veliki broj istraživača da pokušaju da razviju metodologiju određivanja veličine projekata na osnovu UML dokumentacije i objektno-orijentisane specifikacije. Metodom tačaka slučajeva korišćenja se kvantifikuje interakcija između korisnika i sistema [Karner, 1993], [Kusumoto *et al.*, 2004], [Clemmons, 2006]. Podaci se nalaze pregledom dokumentacije slučajeva korišćenja [Rumbaugh, 1994], [Cockburn, 2001], [Constantine and Lockwood, 1999] i modela slučajeva korišćenja prikazanom na slici 23.



Slika 23. Prikaz interakcije korisnika i sistema modelom slučajeva korišćenja.

Karakteristike sistema od značaja za određivanje funkcionalne veličine metodom tačaka slučajeva korišćenja su korisnici sistema i slučajevi korišćenja. Korisnici sistema se pronalaze i kategorizuju na:

1. Jednostavne korisnike koji predstavljaju eksterne sisteme koji komuniciraju sa sistemom po već definisanom protokolu ili API-ju. Ovakvim korisnicima se dodeljuje težina 1.
2. Srednje korisnike koji predstavljaju osobe ili eksterne sisteme koji komuniciraju sa sistemom koji se analizira preko protokola koji je definisan ali se ne može predvideti scenario korišćenja sistema ili sekvenca komandi koje će poslati korisnik. Ovi korisnici imaju težinu 2.
3. Složene korisnike koji predstavljaju osobe koje koriste sistem preko grafičkog interfejsa. Za njih je često potrebno implementirati reagovanje na događaje, validaciju unosa, složenu logiku, interaktivni grafički prikaz sa brojnim dinamičkim elementima, i sl. Ovi korisnici imaju težinu 3.

Pored korisnika sistema analiziraju se i slučajevi korišćenja. Složenost slučajeva korišćenja se određuje brojem transakcija koje su izvršene u okviru slučaja korišćenja. Transakcija je bilo koja aktivnost kojom se vrši razmena poruka između korisnika i sistema. Slučajevi korišćenja se na osnovu broja transakcija klasifikuju u tri kategorije:

1. Jednostavne slučajeve korišćenja koji imaju 3 ili manje transakcija. Ovim slučajevima korišćenja se dodeljuje težinski faktor 5.
2. Srednje slučajeve korišćenja sa 4 do 7 transakcija. Ovim slučajevima korišćenja se dodeljuje težinski faktor 10.
3. Komplikovane slučajeve korišćenja koji imaju više od 8 transakcija. Težinski faktor za ove transakcije je 15.

Veličina sistema se može predstaviti šesto-dimenzionalnim vektorom $\vec{U} = (A_L, A_M, A_H, UC_L, UC_M, UC_H)$ čije komponente predstavljaju broj jednostavnih, srednjih i složenih korisnika i slučajeva korišćenja u sistemu. Na osnovu ove vektorske veličine se mogu definisati norma i metrika sistema na osnovu formula (30) i (31).

$$UUCP = \|U\| = \vec{U} * \vec{w}, \text{ gde je } \vec{w} = (1,2,3,5,10,15) \quad (30)$$

$$d(\vec{U}_1, \vec{U}_2) = \|\vec{U}_1 - \vec{U}_2\| \quad (31)$$

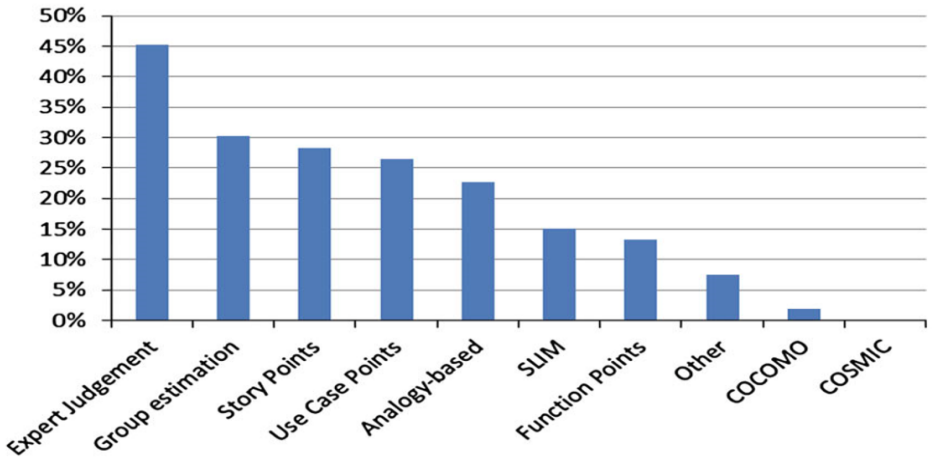
Norma vektora, koja se dobija skalarnim množenjem vektora veličine težinskim vektorom w, predstavlja funkcionalnu veličinu, ili broj neprilagođenih tačaka slučajeva korišćenja (*Unadjusted Use Case Points – UUCP*) kako se ova veličina naziva u UCP terminologiji. U ovu meru sistema ne ulaze dodatne informacije o tehničkim karakteristikama i faktorima okruženja.

Iako UCP metoda nije zvanično standardizovana u vidu nekog ISO standarda, ona je trenutno najčešće korišćena merna metoda koja se koristi za procenu napora [Kassab *et al.*, 2014], [Usman *et al.*, 2014]. Zbog toga je veliki broj istraživača evaluirao ovu metodu [Alves *et al.*, 2013], [Subriadi and Ningrum, 2014], [Lavazza and Robiolo, 2010], [Lavazza and Robiolo, 2012], [Ochodek and Nawrocky, 2010], [Ochodek *et al.*, 2011] i generalni zaključak je da se UCP metodom dobijaju greške procene između 20% i 35%. Najbolje rezultate je dobio Karol [Carroll, 2005] sa greškom ispod 10%

Veliki broj istraživača je istražio različite metode unapređivanja tačnosti UCP metode korišćenjem nelinearnih modela ili veštačke inteligencije [Nassif *et al.*, 2010], [Nassif *et al.*, 2011], [Nassif, 2012], [Nassif *et al.*, 2012], [Azzeh, 2013], [Nassif *et al.*, 2013], [Urbanek *et al.*, 2014], itd. Detaljan pregled i poređenje metoda u ovoj oblasti se može naći u [Kamal *et al.*, 2011].

3.3. Primena softverskih mera u praksi

U [Kassab *et al.*, 2014] je sprovedeno globalno istraživanje o metodama koje se koriste radi procene napora. Rezultati su prikazani na slici 24 [Kassab *et al.*, 2014].



Slika 24. Metode procene napora koje se primenjuju u praksi.

Kao što se može primetiti, najčešće se koristi ekspertska procena, potom slede procena tima, tačke korisničkih priča (engl. *Story Points*) [Schwaber and Beedly, 2001], tačke slučajeva korišćenja i analogija. Metoda funkcionalnih tačaka se koristi u manje od 15% projekata, a ostale metode se ređe koriste.

Od formalnih metoda samo tačke slučajeva korišćenja i funkcionalne tačke korišćene u praksi. Iako je COSMIC metoda dugo najavljivana kao metoda koja će preovladati u domenu merenja funkcionalnosti, ipak nije uzela primat u odnosu na ostale metode. Ostale metode nisu naišle na širu primenu. U nastavku će biti evaluirane UCP, FPA, kao i varijante FPA metode za procenu funkcionalne veličine softvera sa stanovišta mogućnosti procene napora dobijenim funkcionalnim veličinama.

4. Metode procene uticaja nefunkcionalnih zahteva i napora

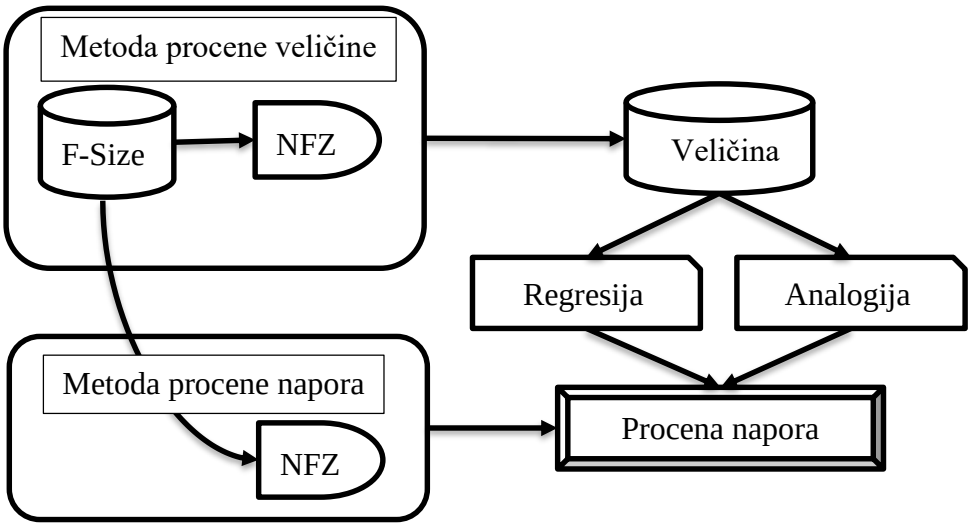
U prethodnom poglavlju je pokazano da se napor potreban radi implementacije projekta može proceniti na osnovu veličina koje kvantifikuju uticaj funkcionalnih zahteva (FZ). Iako napor uglavnom zavisi od funkcionalnih zahteva (slučajeva upotrebe, opisa procesa, itd.), ne može se zanemariti uticaj ne-funkcionalnih zahteva (NFZ) koji ne opisuju šta će softver raditi, nego kako će raditi. Nefunkcionalni zahtevi definišu zahtevane performanse, eksterne interfejse, količine podataka sa kojima će se raditi, ograničenja, itd. Formalno, postupak procene može biti predstavljen kao funkcionalna zavisnost između napora i uticaja funkcionalnih i nefunkcionalnih zahteva, kao što je prikazano u formuli (32).

$$\textit{Procena napora} = f(\text{uticaj(FZ)}, \text{uticaj(NFZ)}) \tag{32}$$

Pravila procene uticaja funkcionalnih zahteva su objašnjena u prethodnom poglavlju. Parametri i pravila koja se koriste za procenu uticaja nefunkcionalnih zahteva u najčešće korišćenim metodama procene su predstavljene u ovom poglavlju.

4.1. Proces procene napora zasnovan na uticaju nefunkcionalnih zahteva

Većina postojećih metoda za procene napora koristi uticaje funkcionalnih i nefunkcionalnih zahteva koji se procesiraju modelima kao što je prikazano na slici 25.



Slika 25. Proces procene napora koji uključuje uticaj nefunkcionalnih zahteva.

Polazna tačka u procesu procene je funkcionalna veličina sistema kojom se na kvantifikuju funkcionalnosti i korisnički zahtevi (označena kao F-Size na slici 25). Ona kvantifikuje funkcionalnosti bez obzira na ne-funkcionalne aspekte, kao što su složenost infrastrukture, timske sposobnosti, potrebna upotrebljivost, performanse sistema, itd.

Metode za procenu veličine mogu da ažuriraju tu veličinu uticajem nefunkcionalnih zahteva čime bi trebalo da se dobije veličina koja bolje kvantifikuje veličinu samog sistema. Ovakva veličina se može koristiti radi procene napora primenom linearne regresije, analogije, neutralnih mreža i drugih metoda. Drugi pristup koji se može primeniti je da se uticaj funkcionalnih zahteva koristi direktno u nekim metodama za procenu napora koje će koristiti sopstvena pravila za procenu napora pomoću veličine i uticaja nefunkcionalnih zahteva. Ovakve metode imaju svoje formula i pravila za kvantifikovanje parametara sistema na osnovu NFZ-ova kao i svoje funkcionalne zavisnosti koje opisuju zavisnost procenjenog napora od veličine.

Trenutno se koriste dva modela za procenu napora prikazana u formulama (33) i (34).

$$\text{Napor} \sim K * \text{Velicina} \tag{33}$$

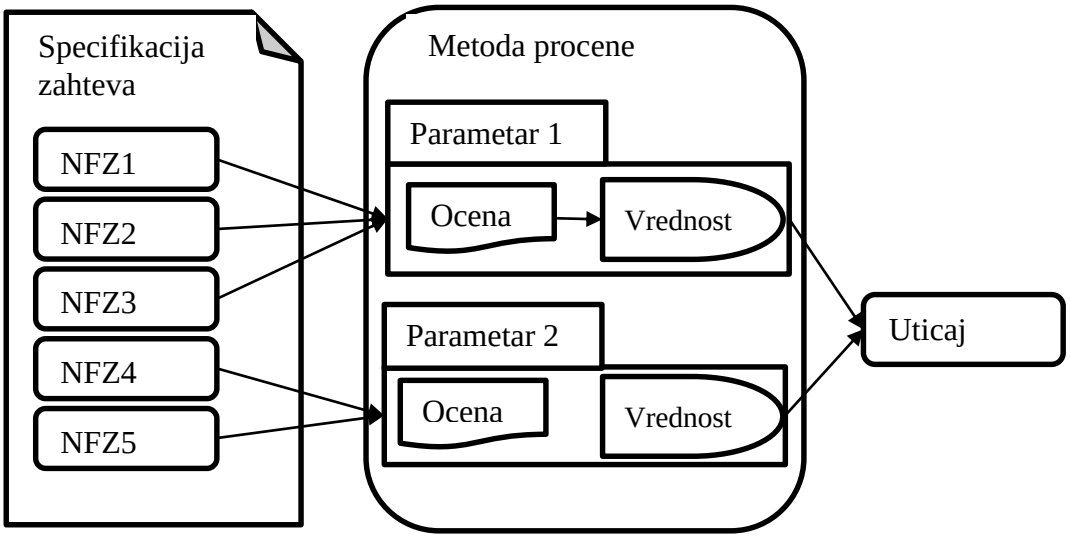
$$\text{Napor} \sim K * \text{Velicina}^{E} \tag{34}$$

Koeficijenti K i E predstavljaju parametre kojima se kvantifikuje uticaj NFZ-ova. U najjednostavnijem slučaju, pretpostavlja se da je napor direktno srazmeran veličini [Briand *et al.*, 1997] tako da uticaj nefunkcionalnih zahteva predstavlja linearni koeficijent koji treba primeniti. U nelinearnim modelima [Putnam, 1997], [Walston and Felix, 1977], [Boehm *et al.*, 2000], [Putnam *et al.*, 2005] pretpostavlja se da nefunkcionalni zahtevi utiču na vrednosti dva parametra, jednog koji služi kao eksponent i drugog koji predstavlja linearni koeficijent. Neke od najčešće korišćenih metoda procene napora su: COCOMO II [Boehm *et al.*, 2000], COBRA [Briand *et al.*, 1997], itd.

4.2. Procena uticaja nefunkcionalnih zahteva

Metode za procenu uticaja NFZ (tj. vrednosti linearnog i eksponencijalnog koeficijenta u formulama (33) i (34) su slične u postojećim metodama procene veličine i napora. Svaka

metoda definiše skup parametara modela (npr. potrebne performanse, složenost proizvoda, itd.) kako bi se klasifikovali srodni nefunkcionalni zahtevi.



Slika 26. Procena uticaja nefunkcionalnih zahteva.

Nefunkcionalni zahtevi se nalaze u specifikacijama zahteva. Sve metode procene definišu konačan broj parametara kojima se NFZ-ovi grupišu tako da više NFZ-ova može da pripada istoj grupi. Grupe se razlikuju od metode do metode ali neki primeri su očekivane performanse, lakoća korišćenja, količina podataka koja treba da se obradi, brzina odziva, prenosivost na druge platforme, sposobnost tima ili pojedinaca u timu i slično.

Svaki parametar se ocenjuje ocenama 0-5, ili mali-veliki na osnovu nefunkcionalnih zahteva grupisanih oko parametra. Ocene se potom kvantifikuju i svaki parametar dobija neku numeričku vrednost. Konačno, na osnovu numeričkih vrednosti parametara se određuje uticaj nefunkcionalnih zahteva, kao što je prikazano na slici 25. Ovaj uticaj se koristi za korigovanje funkcionalne veličine (F-Size) kako bi se dobila reprezentativnija veličina sistema, tako što se koristi kao linearni koeficijent ili eksponent u formulama (33) ili (34).

4.3. Parametri za ocenjivanje uticaja nefunkcionalnih zahteva

U ovoj sekciji će biti predstavljena pravila za procenu uticaja nefunkcionalnih zahteva u reprezentativnim metodama procene veličine i napora.

4.3.1. Metoda funkcionalnih tačaka

Kao što je opisano u prethodnom poglavlju, FPA [IFPUG, 1994] se koristi radi određivanja funkcionalne veličine sistema. Ova veličina se naziva neprilagođena veličina u FPA terminologiji zato što se dodatno prilagođava uticajem nefunkcionalnih zahteva kako bi se dobila veličina u funkcionalnim tačkama. U FPA metodi se koristi 14 parametara poznatih kao globalne karakteristike sistema (engl. *Global System Characteristic* - GSC), prikazanih u tabeli 7.

Tabela 7. Globalne karakteristike sistema koje se ocenjuju po FPA metodi.

GSC	Parametar
1	Komunikacija podataka (engl. *Data communication*): Koliko komunikacioni objekti mogu da pomognu u prenosu ili razmeni informacija?
2	Distribuirana obrada podataka (engl. *Distributed data processing*): Postoji li distribuirana obrada podataka?
3	Performanse (engl. *Performance*): Koliko su performanse bitne korisnicima?
4	Intenzivno korišćenje konfiguracije (engl. Heavily used configuration): Koliko se koristi hardverska platforma i infrastruktura u kojoj radi softver?
5	Transakciona obrada (engl. Transaction rate): Koliko se često se izvršavaju transakcije u sistemu?
6	Direktan unos podataka (engl. On-Line data entry): Koliko procenata podataka se unosi direktno?
7	Efikasnost krajnjeg korisnika (engl. End-user efficiency): Koliko je bitno da korisnik efikasno radi sa softverom?
8	Direktno ažuriranje (engl. On-Line update): Koliko podataka se direktno ažurira?
9	Kompleksnost obrade (engl. Complex processing): Koliko je kompleksna obrada podataka u sistemu?
10	Upotrebljivost (engl. Reusability): Koliko je bitna ponovna upotrebljivost koda?
11	Lakoća instalacije (engl. Installation ease): Koliko je bitna lakoća instalacije?
12	Lakoća upotrebe (engl. Operational ease): Koliko je bitna lakoća upotrebe?
13	Razuđenost na više lokacija (engl. Multiple sites): Da li je projektni tim podeljen na više lokacija?
14	Spremnost na promene (engl. Facilitate change).

Globalne karakteristike sistema se procenjuju na osnovu NFZ koji se odnose na njih, i celobrojna ocena u opsegu od 0 do 5 se dodeljuje svakoj od njih. Vrednosti ocena se direktno koriste za određivanje uticaja nefunkcionalnih zahteva pomoću formule (35).

$$VAF = 0.65 + 0.01 * \sum_{i=1}^{14} GSC_i \tag{35}$$

VAF parametar predstavlja linearni koeficijent kojim se množi funkcionalna veličina. Konstanta 0,65 je odabrana kako bi uticaj nefunkcionalnih zahteva bio 1 kada bi svi GSC parametri imali prosečne vrednosti.

4.3.2. Metoda klasnih tačaka

Metodom klasnih tačaka (CP) [Costangiola and Tortora, 2005] se procenjuje veličina sistema na osnovu složenosti klasa i UML modela. Slično FPA metodi, u CP metodi se koristi 24 parametara za modifikovanje klasne veličine sistema. Prvih 14 parametara su identični GSC parametrima u FPA metodi [IFPUG, 1994], dok je dodato 10 novih parametara: prilagodljivost korisnika (C15), brzo pravljenje prototipa (C16), Interaktivnost (C17), postojanje više interfejsa (C18), efikasnog upravljanja (C19), sposobnost programera (C20), bezbednost (C21), pouzdanost (C22), održavanje (C23) i prenosivost na druge platforme (C24). Ovi parametri se ocenjuju isti način kao u FPA metodi (tj. u opsegu od 0 do 5). Uticaj nefunkcionalnih zahteva se izračunava pomoću formule (36).

$$VAF = 0.55 + 0.01 * \sum_{i=1}^{24} C_i \tag{36}$$

Formula je veoma slična FPA formuli uz razliku da se koristi konstanta 0.55 koja postavlja vrednost na 1 kada svih 24 parametara imaju podrazumevane (prosečne) vrednosti.

4.3.3. Metoda tačaka slučajeva korišćenja

U metodi tačaka slučajeva korišćenja (UCP) [Karner, 1993] se koristi 21 parametar radi ažuriranja UUCP veličine - 13 predstavljaju tehničke karakteristike (Ti) i 8 predstavljaju faktore okruženja (Ei) prikazane u tabeli 8.

Slično FPA i CP metodama, parametri se ocenjuju vrednostima od 0 do 5 u zavisnosti od procene odgovarajućih nefunkcionalnih zahteva. Međutim, za razliku od ovih metoda, parametri u UCP metodi nemaju identičan stepen uticaja na konačne veličine.

Tabela 8. Faktori koji se ocenjuju u UCP metodi.

ID	Parametar	wt_i	ID	Parametar	we_i
T_1	Distribuirani sistem	2.0	E_1	Familijarnost sa korišćenim razvojnim procesom	1.5
T_2	Vreme odgovora - performanse	1.0			
T_3	Efikasnost krajnjeg korisnika	1.0	E_2	Iskustvo u domenu aplikacije	0.5
T_4	Kompleksnost internog procesa	1.0	E_3	Iskustvo u objektno-orijentisanim tehnologijama	1.0
T_5	Ponovno iskorišćenje koda	1.0	E_4	Sposobnost glavnog analitičara	0.5
T_6	Lakoća instalacije	0.5			
T_7	Lakoća korišćenja	0.5	E_5	Motivacija tima	1.0
T_8	Prenosivost na ostale platforme	2.0	E_6	Stabilnost zahteva	2
T_9	Održavanje sistema	1.0	E_7	Članovi tima koji ne rade puno radno vreme	-1.0
T_{10}	Konkurentnost – paralelno procesiranje	1.0	E_8	Složenost programskog jezika	-1.0
T_{11}	Sigurnosni zahtevi	1.0			
T_{12}	Pristup eksternih sistema	1.0			
T_{13}	Treniranje krajnjih korisnika	1.0			

Svaki parametar ima svoju težinu prikazanu u tabeli 8, kako bi se napravila razlika među parametrima koje imaju veći uticaj na napor. Uticaj nefunkcionalnih zahteva se određuje pomoću formule (37).

$$VAF = (0.6 + 0.01 * \sum_{i=1}^{13} wt_i * T_i) * (1.4 - 0.03 * \sum_{j=1}^{8} we_j * E_j \quad (37)$$

Ovaj parametar se koristi kao linearni koeficijent kojim se ažurira funkcionalna veličina određena na osnovu korisnika i slučajeva korišćenja u sistemu. Slično formulama (35) i (36), koeficijenti u formuli (37) su odabrani tako da vrednost VAF parametra bude 1 kada svi parametri imaju prosečne vrednosti.

4.3.4. COBRA

Brajand [Briand *et al.*, 1997] je predložio COBRA (engl. *COst estimating, Benchmarking and Risk Assessment*) model kojim se napor procenjuje linearnom formulom (33) na osnovu sledećih parametara:

1. Veličine projekta izražene u bilo kojoj jedinici mere (funkcionalne tačke, tačke slučajeva korišćenja, itd.),
2. Produktivnosti koja je obrnuto proporcijalna linearnom koeficijentu korišćenom u formuli (33).

Produktivnost se izražava kao količina softvera koju projektni tim može implementirati u jedinici vremena. Karnerov odnos napora i broja tačaka slučajeva korišćenja od 20 sati rada po tački [Karner, 1993] se može shvatiti kao vrednost obrnuto srazmerna produktivnost u COBRA terminologiji.

Osnovna pretpostavka COBRA metode je da svaka softverska organizacija može pronaći standardni projekat koji služi kao referenca za procene na novim projektima. „Standardni projekat“ je idealizovani projekat koji organizacija može da implementira i za koji se znaju performanse tima. Za ovakav standardni projekat se može odrediti nominalna produktivnost na osnovu istorijskih podataka o sličnim projektima koji su već implementirani.

Potom se evaluiraju parametri modela i procenjuje koliko se razlikuju na novom projektu u odnosu na standardni projekat. Svaka razlika može da smanji ili poveća produktivnost na novom projektu. Vrednost procenjene produktivnosti na projektu se koristi u formuli (40) kako bi se procenio napor.

Brajandova metoda ne unosi veće novine u pogledu određivanja funkcionalne zavisnosti pošto i ostale metode uvođenjem ocene kompleksnosti sistema implicitno pretpostavljaju linearnu zavisnost veličine i napora. Pored toga nije pokazano da ova metoda daje bolje rezultate od jednostavne interpolacije koja se koristi češće, a uz to se i lakše može opravdati.

4.3.5. COCOMO II model

COCOMO II [Boehm *et al.*, 2000] je model kojim se na osnovu broja linija koda ili funkcionalne veličine sistema i skupa parametara kojima se opisuje NFZ uticaj procenjuje napor na projektu. COCOMO II koristi formulu (34) u kojoj se veličina ažurira linearnim koeficijentom i eksponentom.

Da bi se odredile vrednosti linearnog koeficijenata i eksponenta, u COCOMO II modelu se koriste 22 parametara koji su podeljeni u dve grupe:

1. Faktori skaliranja (engl. Scale Factors – SF) koji služe da se odredi eksponent u formuli (34),
2. Multiplikatori napora (engl. Effort multipliers – EM) koji služe da se odredi linearni koeficijent u formuli (34).

Multiplikatori napora koji utiču na linearni koeficijent su prikazani u tabeli 9. U COCOMO II modelu postoji 17 multiplikatora napora.

Tabela 9. Multiplikatori napora koji se ocenjuju u COCOMO II modelu.

Parametar	Opis
RELY	Pouzdanost (engl. Required Software Reliability).
DATA	Veličina baze (engl. Data Base Size).
CPLX	Kompleksnost proizvoda (engl. Product Complexity): kontrolne operacije, izračunavanja, operacije koje zavise od uređaja, operacije upravljanja podacima, operacije upravljanja korisničkim interfejsom.
RUSE	Zahtevana ponovna upotrebljivost (engl. Required Reusability): Kvantifikuje količinu dodatnog napora koji je potrebno uložiti kako bi se razvijene komponente iskoristile u drugim projektima.
DOCU	Dokumentacija (engl. Documentation Match to Life-cycle Needs): Koliko je bitno da dokumentacija prati proizvod u svim životnim ciklusima projekta?
TIME	Vreme (engl. Time Constraint): Koliko je bitno vreme izvršavanja?
STOR	Skladištenje podataka (engl. Storage Constraint): Postoje li neka ograničenju u vidu skladištenja podataka?
PVOL	Promenljivost platforme (engl. Platform Volatility): Uticaj promenljivosti hardverske ili softverske platforme na sistem.
ACAP	Sposobnost analitičara (engl. Analyst Capability).
PCAP	Sposobnost programera (engl. Programmer Capability).
AEXP	Iskustvo sa aplikacijom (engl. Applications Experience).
PEXP	Iskustvo na platformi (engl. Platform Experience).
LTEX	Iskustvo u radu sa programskim jezikom i alatima (engl. Language and Tool Experience).
PCON	Kontinuitet tima (engl. Personnel Continuity).
TOOL	Korišćenje softverskih alata (engl. Use of Software Tools).
SITE	Geografska razuđenost tima (engl. Multi-Site Development).
SCED	Zahtevani raspored projektnih zadataka (engl. Required Development Schedule).

U COCOMO II modelu se koristi šest faktora skaliranja prikazanih u tabeli 10.

Tabela 10. Faktori skaliranja koji se ocenjuju u COCOMO II modelu.

Parametar	Opis
PREC	Razumevanje organizacije ciljeva projekta i potrebnih tehnologija.
FLEX	Fleksibilnost razvoja (engl. Development Flexibility): Predstavlja stepen slobode rada u odnosu na zahteve, ograničenja i eksterne standarde.
RESL	Arhitektura i rešavanje rizika (engl. Architecture and Risk Resolution): Kvantifikuje razumevanje softverske arhitekture kvalitet kao i količinu nerešenih rizika.
TEAM	Kohezija tima (engl. Team cohesion): Kvantifikuje spremnost svih članova tima i eksternih faktora da rade zajedno kao tim.
PMAT	Zrelost procesa (engl. Process Maturity): Zrelost procesa koji se koristi u razvoju zasnovano na CMM-CMMI modelu za ocenjivanje zrelosti procesa.

Parametri se opisno kvalifikuju vrednostima u opsegu od "veoma nizak" do "ekstra visok". Svakoj opisnoj vrednosti je dodeljena odgovarajuća kvantitativna vrednost na osnovu tabele 11 [Boehm *et al.*, 2000]. Puna tabela je prikazana u prilogu.

Tabela 11. Primer težinskih faktora koji se dodeljuju COCOMO II parametrima.

Parametar	Simbol	VL	L	N	H	VH	XH
PREC	SF_1	0.05	0.04	0.03	0.02	0.01	0
FLEX	SF_2	0.05	0.04	0.03	0.02	0.01	0
RESL	SF_3	0.05	0.04	0.03	0.02	0.01	0
DOCU	EM_5	0.85	0.93	1	1.08	1.17	
PCAP	EM_{10}	1.37	1.16	1	0.87	0.74	
PCON	EM_{11}	1.26	1.11	1	0.91	0.83	
AEXP	EM_{12}	1.23	1.1	1	0.88	0.8	

Nominalna vrednost svakog multiplikatora napora se preslikava na 1.00. Napor se procenjuje korišćenjem formule (38).

$$Effort = 2.94 * \prod_{i=1}^{17} EM_i * Size^{0.91+0.01*\sum_{j=1}^{5} SF_j} \quad (38)$$

Parametri i težine prikazane u tabeli 11 su kalibrisane na velikom skupu podataka. Međutim, to ne garantuje da su te vrednosti optimalne i na novim projektima.

4.3.6. Metoda Veb objekata

Sve veći razvoj interneta povećao je potrebu za prilagođenim metodama koje se mogu primeniti na internet aplikacije. Donald Reifer je predložio metodu Veb objekata za procenu napora kojom se određuje veličina sistema na osnovu parametara karakterističnih za internet aplikacije kao što su skriptovi ili multimedijalni fajlovi [Reifer, 2000]. Ova veličina se modifikuje parametrima koji zavise od nefunkcionalnih zahteva. U ovoj metodi se koristi modifikovani podskup parametara iz COCOMO II modela. Parametri označeni sa WCD (engl. *Web Cost Drivers*) su:

1. **RCPX** – Pouzdanost i kompleksnost proizvoda (engl. *Product Reliability & Complexity*) koji kombinuje COCOMO II parametre RELY, DATA, CPLX i DOCU
2. **PDIF** – složenost platforme (engl. *Platform Difficulty*) koji kombinuje COCOMO II parametre TIME, STOR i PVOL
3. **PERS** – sposobnost tima (engl. *Personnel Capability*) koji kombinuje COCOMO II parametre ACAP, PCAP i PCON
4. **PREX** – iskustvo tima (engl. *Personnel experience*) koji kombinuje parametre APEX, PLEX i LTEX
5. **FCIL** – Sredstva (engl. *Facilities*) koji kombinuje COCOMO II parametre TOOL i SITE
6. **SCED** – Zahtevani raspored projektnih zadataka i rokovi (engl. *Required development schedule*) koji je direktno preuzet iz COCOMO II modela.

Slično COCOMO II modelu, svaki parametar se ocenjuje vrednostima od "veoma nizak" do "ekstra visok" i odgovarajuća numerička vrednost je dodeljena svakoj opisnoj vrednosti. Svaka "nominalna" vrednost se preslikava na 1,00 a procena napora se određuje formulom (39).

$$Effort = A * \prod_{i=1}^{8} WCD_i * Size^P \tag{39}$$

Konstanta A može imati vrednosti od 2.0 do 2.7, u zavisnosti od vrste veb aplikacije, a eksponent P je konstanta (1.05) koja se primenjuje samo ako je aplikacija iz domena elektronske trgovine ili finansijska.

4.4. Zaključak

Kao što se može primetiti, parametri kojima se procenjuje NFZ uticaj u različitim metodama su veoma slični, tako da je teško proceniti koji metod je najprimenljiviji. U nekim konkretnim slučajevima parametri u postojećim metodama mogu biti neprimenljivi (npr. lakoća instalacije u internet aplikacijama koje se samo otvaraju u pregledačima). Nekada se određeni parametri mogu ignorisati bez većeg uticaja na tačnost procena uz pojednostavljene modele [Ochodek *et al.*, 2011]. Pored toga, možemo da imamo neke parametre koji bi mogli biti od značaja za procenu napora na projektima koji nisu uključeni u metodu procene napora koja bi se koristila za taj projekat. Imajući ovo u vidu, nekoliko istraživača je modifikovalo postojeće metode kako bi dobili dodatna poboljšanja procena. Ochodek [Ochodek *et al.*, 2011] je smanjio broj UCP parametara sa 21 na 6, Frohnhoff i Engels [Frohnhoff and Engels, 2008] su uveli novi skup parametara, Rue koji je definisao novu Veb-COBRA metodu zasnovanu na COBRA modelu prilagođenom za internet projekte [Ruhe *et al.*, 2003]. Zbog toga je bitno ispitati kako se postojeći parametri mogu kombinovati kako bi se poboljšale procene.

II Deo

U drugom delu teze će biti upoređene postojeće metode za procenu veličina softverskih projekata na osnovu funkcionalnih zahteva, kao i metode za procenu uticaja nefunkcionalnih zahteva. Potom će biti predstavljeni predlozi za kombinovanje i unapređivanje postojećih metoda procena uticaja funkcionalnih i nefunkcionalnih zahteva kako bi se identifikovale optimalne metode procene napora.

Prvi korak u istraživanju je komparativna analiza metoda za procenu veličine softvera na osnovu funkcionalnosti. Cilj analize je identifikovanje metoda procene veličine koje se mogu uspešno primenjivati radi procene napora na projektima.

Slična analiza je primenjena nad najčešće primenjivanim metodama procene uticaja nefunkcionalnih zahteva.

Potom su predstavljeni načini kako se postojeće metode mogu kombinovati i unapređivati kako bi se dobile nove metode kojima se napor na projektu može tačnije proceniti.

5. Komparativna evaluacija metoda procene funkcionalnih veličina korišćenih radi procene napora

U ovoj sekciji će biti istraženo da li se metode ocenjivanja funkcionalne veličine softvera mogu koristiti radi procene napora i sa kolikom pouzdanošću. U istraživanju se posmatraju sledeće faze projekta:

1. Rane faze projekta gde se na osnovu delimično poznatih i nepotpunih zahteva moraju dati okvirne procene.
2. Faza analize gde su već dobro definisani i dokumentovani zahtevi i gde se mogu vršiti procena sa mnogo više informacija koje su na raspolaganju.

Cilj ovog dela istraživanja je provera da li se metode procene veličine softvera mogu koristiti u ovim fazama radi procene napora i sa kolikom preciznošću procena.

5.1. Funkcionalne veličine koje se mogu primeniti tokom ranih faza projekta

Rane faze projekta karakteriše ograničen broj informacija koje su na raspolaganju. Dokumenti koji su na raspolaganju radi procene veličine su:

1. Dokument vizije projekta u kome su okvirno opisani korisnici sistema, glavne funkcionalnosti, slučajevi korišćenja, itd.
2. Konceptualni modeli podataka ili poslovnih procesa u kome se mogu naći glavni poslovni entiteti i procesi.

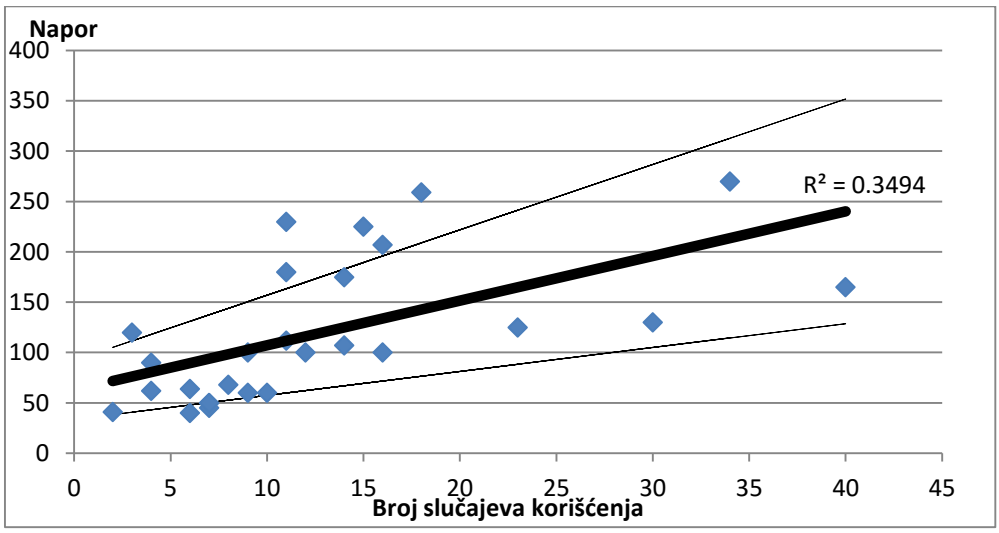
Iako nema mnogo informacija u ranim fazama projekta, veoma je bitno dati što je moguće bolje procene pošto one najviše utiču na tok projekta. U ovom poglavlju će biti prikazane metode koje mogu pomoći prilikom procene na osnovu nedovoljno dobro definisanih zahteva.

5.1.1. Procena napora prebrojavanjem slučajeva korišćenja

S obzirom da se u ranijim fazama projekta identifikuju slučajevi korišćenja ali se ne opisuju sa dovoljnim brojem detalja kako bi se mogla primeniti standardna metoda tačaka slučajeva korišćenja [Reifer, 2000], potrebno je koristiti aproksimativnu meru. Kao mera veličine sistema može primeniti broj identifikovanih slučajeva korišćenja. Izmerena veličina sistema se može razlikovati od konačne mere izražene stvarnim tačkama

slučajeva korišćenja ali imajući u vidu oskudnost informacija ova odstupanja mogu biti prihvatljiva.

U skupu podataka koji je korišćen u analizi ukupno 28 projekata ima dovoljno informacija na osnovu kojih se mogu identifikovati i prebrojati slučajevi korišćenja. Za ove projekte je pronađen broj identifikovanih slučajeva korišćenja i upoređen sa vremenom potrebnim da se implementira svaki od projekata. Na slici 27 je prikazan napor potreban za izradu sistema u zavisnosti od broja slučajeva korišćenja (NUC). Analiza je izvedena na skupu od 28 projekata iz referentnog skupa veličine od 50 do 300 čovek-dana. Pored funkcionalne zavisnosti, na slici je prikazana i oblast koja obuhvata projekte čija je greška procene manja od 25%.



Slika 27. Procena napora na osnovu broja slučajeva korišćenja.

Stepen korelacije između veličine izražene u broju slučajeva korišćenja i veličine izražene u čovek-dan jedinici iznosi 58%, što nije preterano dobar rezultat. U slučaju da se koristi linearna regresija radi procene napora na osnovu veličine najbolja zavisnost koja se može dobiti je prikazana formulom (40).

$$Effort = 4{,}44 * NUC + 62{,}8 \quad (40)$$

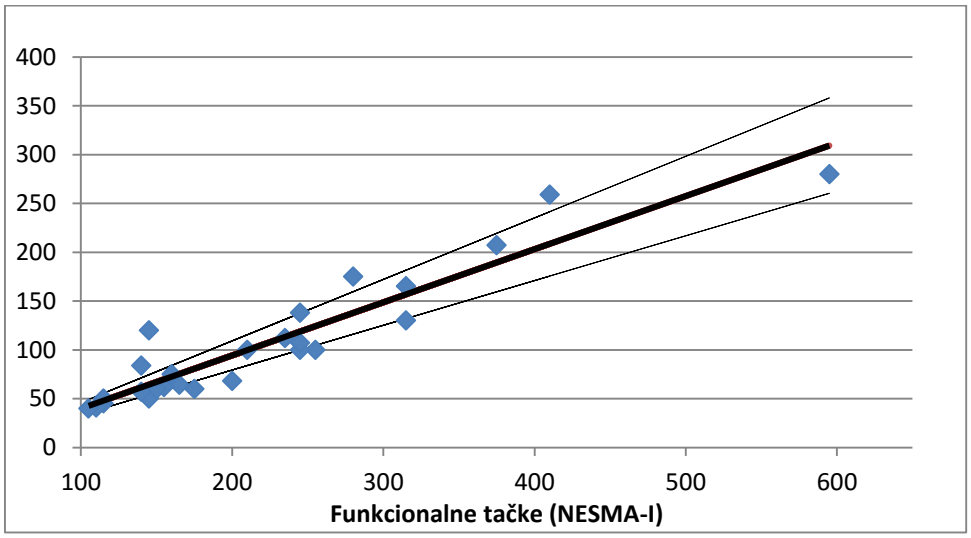
U formuli je napor izražen u danima, dok je NUC broj slučajeva korišćenja. Srednja greška koja se može dobiti ovakvom funkcionalnom zavisnošću je 46%. Maksimalna

greška procene je 126%, što i nije loše imajući u vidu Boemove [Boehm *et al.*, 2000] procene o pouzdanosti na početku projekta (150%), ali su u istraživanju nađene metode koje mogu dosta preciznije odrediti vreme potrebno za implementaciju na osnovu drugih veličina.

Smit [Smith, 2003] je izvršio sličnu analizu procene napora na osnovu broja slučajeva korišćenja sa malo boljim rezultatima. Međutim, analizirani sistemi u njegovom slučaju su imali konzistentne slučajeve korišćenja sa detaljnom analizom formata slučajeva korišćenja. U referentnom skupu slučajevi korišćenja nisu bili standardizovani – čak u nekim slučajevima potpuno nekonzistentni što može objasniti veoma lošu funkcionalnu zavisnost između veličine i napora.

5.1.2. Procena napora NESMA indikativnom metodom

NESMA indikativna metoda (NESMA-I) se može primeniti na 26 projekata u skupu projekata koji se koristi u istraživanju. Primenom linearne regresije na skup tih projekata veličine od 50 do 300 čovek-dana dobijen je grafik zavisnosti napora od broja funkcionalnih tačaka po NESMA-I metodi koji je prikazan na slici 28. Na slici je prikazana i oblast grešaka manjih od 25%.



Slika 28. Procena napora na osnovu veličine određene NESMA indikativnom metodom.

Koeficijent R^2 je 0.86 a koeficijent korelacije 0.93 tako da ova veličina može verno da pruži informacije o naporu potrebnom za implementaciju sistema. Na slici je prikazana i

oblast koja predstavlja granice projekata čije je odstupanje manje od 25%. Primenom linearne regresije dobija se funkcionalna zavisnost prikazana u formuli (41).

$$Effort = 0{,}54 * UFP - 14{,}45 \quad (41)$$

UFP predstavlja veličinu sistema izraženu u funkcionalnim tačkama određenim NESMA indikativnom metodom. Primenom ove funkcionalne zavisnosti dobija se srednja greška od 16%, uz maksimalnu grešku od 46%. Pouzdanost je dosta bolja od Boemovih (150%) i PMI (50%) rezultata.

5.1.3. Pregled rezultata

Tokom faze upoznavanja sa projektom, projektni tim ne raspolaže sa velikim brojem informacija o samom sistemu, ali i pored toga mora da odredi bar okvirne procene vremena potrebnog za izradu.

U tabeli 12 su prikazane mere koje se mogu odrediti tokom faze upoznavanja sa projektom kao i stepen njihove korelacije sa naporom na osnovu analize sprovedene nad referentnim skupom podataka.

Tabela 12. Pouzdanost metoda procene koje se mogu koristiti u inicijalnoj fazi.

	Parametri modela			
Tehnika merenja	R	MMRE	MRE_{max}	PRED(25)
NESMA indikativna	93%	16%	46%	0.8
Broj slučajeva korišćenja	58%	46%	126%	0.4

NESMA indikativna metoda se može koristiti radi procene veličine softvera i kreiranja inicijalnih planova izrade projekta. Greške procene su u skladu sa PMI i Boemovim kriterijumima gde se greške kreću od 50% po PMI rezultatima (PMI-I) do 150% po Boemu (Boem-RS).

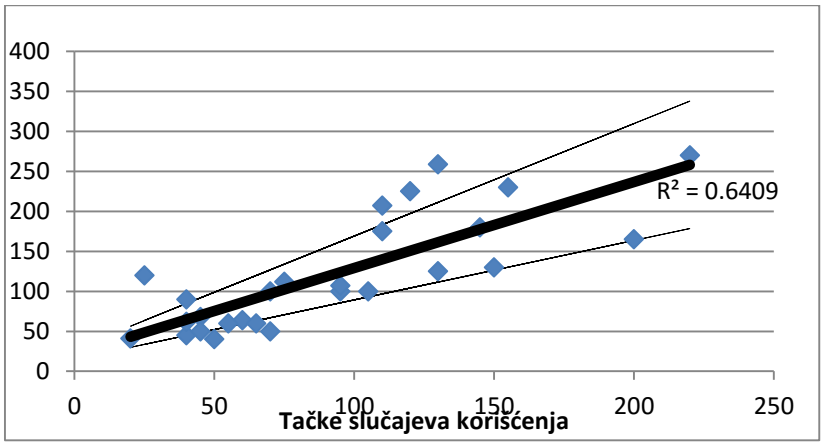
5.2. Funkcionalne veličine koje se mogu primeniti tokom elaboracije projekta

Elaboracija je faza u životnom ciklusu projekta u kojoj se detaljno definišu softverski zahtevi i funkcionalna specifikacija. Tokom elaboracije se mogu koristiti tačke slučajeva korišćenja, NESMA procenjena varijanta metode funkcionalnih tačaka, Mark II metoda i

standardna metoda funkcionalnih tačaka. U narednim sekcijama će biti prikazano koliko efikasno se mogu koristiti ove funkcionalne veličine radi procene napora projekta.

5.2.1. Tačke slučajeva korišćenja

U skupu referentnih podataka 27 projekata ima definisan model slučajeva korišćenja. S obzirom da se u ovom delu istraživanja analizira mogućnost primene funkcionalne veličine radi procene napora, nisu određivani nefunkcionalni parametri tako da je korišćena neprilagođena veličina tačaka slučajeva korišćenja (UUCP) kao mera na osnovu koje je procenjen napor na projektima. Model dobijen linearnom regresijom nad skupom parova veličina i napora je prikazan na slici 29.



Slika 29. Procena napora na osnovu tačaka slučajeva korišćenja.

Funkcionalna zavisnost dobijena linearnom regresijom je prikazana u formuli (42):

$$\text{Effort} = 1{,}08 * \text{UUCP} + 21{,}67 \qquad (42)$$

U regresionom modelu je dobijen koeficijent R^2 od 0.64. Srednja relativna greška na poznatom skupu podataka je 31% mada za neke projekte relativna greška dostiže i 94%. Samo 40% grešaka procene je manje od 25% što nije preterano dobar rezultat (ova oblast je označena ne slici 29). Po Boemovim i PMI podacima greška u ovoj fazi može da se kreće od 110% do 50% što znači da ova metoda može da uđe u traženi opseg tačnosti, međutim druge metode koje su korišćene radu daju znatno bolje rezultate.

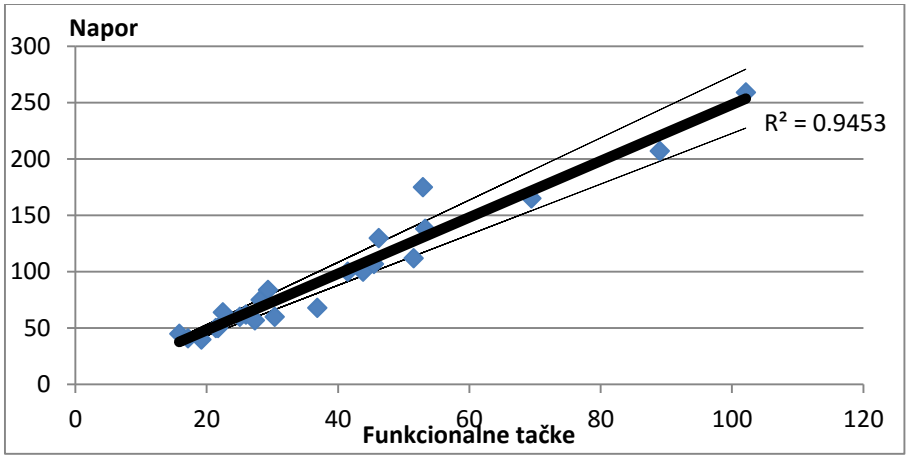
Osnovni problem u posmatranom skupu projekata (koji se može manifestovati i u opštem slučaju) je nepostojanje definicija i standarda za pisanje slučajeva korišćenja. Primećene

su velike razlike u načinu identifikacije i pisanja slučajeva korišćenja što veoma zavisi od analitičara koji je pisao slučajeve korišćenja. Identifikovani su neki scenariji slučajeva korišćenja u kojima bi se slučaj korišćenja iz jednog projekta mogao predstaviti sa nekoliko slučajeva korišćenja u drugom što bi značajno promenilo veličinu sistema.

U literaturi se ne može naći jedinstveni standard za pisanje slučajeva korišćenja. Kokburn [Cockburn, 2001] definiše pet različitih oblika slučajeva korišćenja, Larman [Larman, 2001] dva oblika – korisničke i sistemske, Konstantin i Lokvud [Constantine and Lockwood, 1999], [Lockwood and Lockwood, 1999] definišu esencijalne slučajeve korišćenja kojima se opisuju samo apstraktne korisničke interakcije, dok Gelperin [Gelperin, 2004] predlaže detaljno pisanje slučajeva korišćenja zasnovano na struktuiranom engleskom jeziku [DeMarco, 1978] u vidu posebnog pseudo-jezika za definisanje funkcionalnosti. Odsustvo standarda za specifikaciju slučajeva korišćenja uzrokuje velike probleme u UCP metodi zato što je način određivanja veličine zavisi od načina kako su dokumentovanje transakcije i slučajevi korišćenja.

5.2.2. Mark II metoda

Na osnovu analize dokumenata i identifikacije potrebnih elemenata potrebnih za merenje Mark II metodom dobijena je funkcionalna zavisnost napora potrebnog da se projekti iz referentnog skupa implementiraju od veličine. Ova funkcionalna zavisnost primenjena na uzorku od 23 projekata veličine od 50 do 250 čovek-dana je prikazana na slici 30.



Slika 30. Procena napora na osnovu Mark II veličine.

R^2 koeficijent je 0.95 što je odličan rezultat imajući u vidu količinu raspoloživih informacija, tako da je moguće odrediti funkcionalnu zavisnost linearnom regresijom kojom se dobija funkcionalna zavisnost prikazana formulom (43).

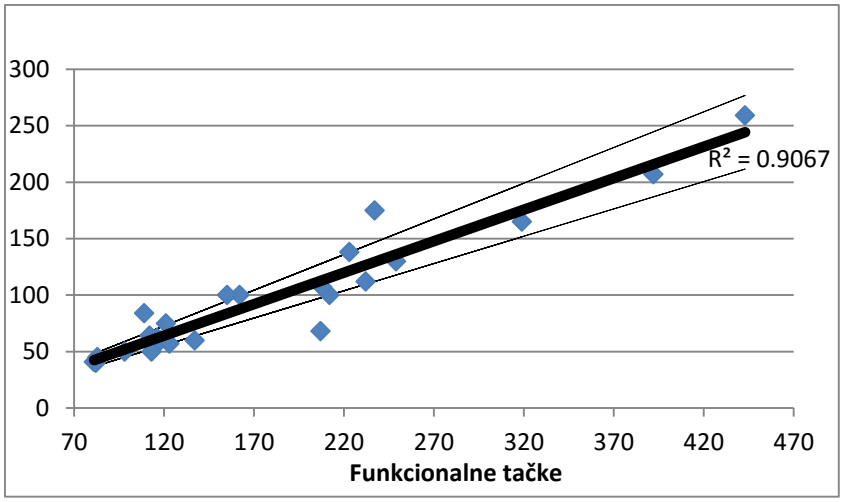
$$\text{Effort} = 2,5 * \text{F-Size} + 1,85 \quad (43)$$

F-Size je veličina sistema određena Mark II metodom. Srednja relativna greška posmatrana na skupu projekata je 10%, uz maksimalnu vrednost od 30% što je mnogo bolje od Boemovih i PMI rezultata koji se kreću od 50% do 110%. Na slici 30 je prikazana i oblast unutar koje su projekti čija je greška procene manja od 10%. Čak 88% grešaka je manje od 25% što je odličan rezultat.

Kao što se vidi rezultati su prilično dobri – samo jedan projekat značajnije odstupa od srednje greške procene. Mark II metoda je sasvim prihvatljiva mera sistema kojom se dobijaju relativno pouzdane mere sistema koje imaju dobar koeficijent korelacije sa naporom potrebnim za realizaciju projekta.

5.2.3. NESMA procenjena metoda

Radi primene NESMA procenjene metode, potrebno je identifikovati fajlove i transakcije u sistemu bez ulaženja u njihove detalje i procenu njihove složenosti. U referentnom skupu podataka 24 projekata imaju dovoljno validne modele baze podataka, analitičke/UML modele, vizije projekata, modele slučajeva korišćenja na osnovu kojih se mogu identifikovati fajlovi i transakcije iako se ne mogu klasifikovati po složenosti. NESMA procenjena metoda je primenjena na ovom podskupu projekata veličine od 50 do 250 čovek-dana i rezultati predikcije linearnom regresijom su prikazani na slici 31.



Slika 31. Procena napora na osnovu NESMA procenjene veličine.

R^2 koeficijent je 0.9 tako da se može izvršiti linearna regresija što se vidi i na slici 31. Linearnom regresijom je dobijena zavisnost prikazana formulom (44).

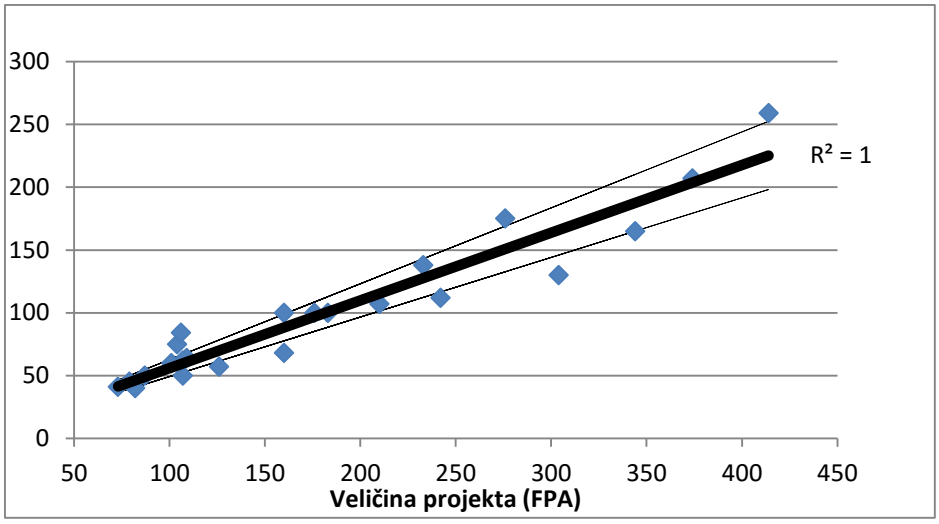
$$\text{Effort} = 0{,}56 * \text{FP} - 2{,}67 \tag{44}$$

Linearna regresija je primenjena na projekte što omogućava predviđanje napora sa srednjom relativnom greškom od 13% iako je maksimalna greška 65%. Gornja i donja granica srednje relativne greške je prikazana na slici 31 i kao što se može videti samo za tri projekta postoje značajna odstupanja. Čak 91% grešaka je manje od 25% što je odličan rezultat.

Pouzdanost NESMA procenjene metode je približno jednaka pouzdanosti Mark II metode opisane u prethodnoj sekciji tako da se obe metode mogu koristiti kao validne mere sistema. Kao i Mark II metoda, NESMA procenjena na skupu referentnih podataka daje mnogo bolje rezultate nego što su rezultati po Boemu i PMI-ju.

5.2.4. Funkcionalne tačke

Iz referentnog skupa projekata su izabrani svi koji imaju detaljnu funkcionalnu specifikaciju tako da se može primeniti standardna metoda funkcionalnih tačaka. Analizom veličine 23 projekata iz referentnog skupa i primenom analize funkcionalnih tačaka dobija se zavisnost napora od veličine izražene u funkcionalnim tačkama prikazana na slici 32.



Slika 32. Procena napora na osnovu funkcionalnih tačaka (FPA).

R^2 koeficijent je 0.92 što znači da funkcionalna veličina određena FPA metodom odlično korelisana sa naporom. Linearnom regresijom je dobijena zavisnost predstavljena formulom (45).

$$\text{Effort} = 0{,}54*\text{UFP} + 2 \tag{45}$$

Ovom funkcionalnom zavisnošću se dobija srednja relativna greška od 12% i maksimalna greška od 20%. 96% grešaka je manje od 25% što je odličan rezultat.

5.2.5. Pregled metoda koje se mogu primenjivati tokom faze elaboracije

Tokom faze elaboracije se može odrediti veličina sistema na osnovu funkcionalnih zahteva, modela i specifikacije. Uporedni rezultati metoda koje su bile primenjene u radu nad referentnim skupom podataka su prikazani u tabeli 13.

Tabela 13. Pregled metoda procene koji se mogu koristiti u fazi elaboracije.

	Parametri predikcionih modela			
Metoda	R	MMRE	MRE_{max}	PRED(25)
Tačke slučajeva korišćenja	80%	31%	94%	0.4
Mark II	92%	13%	65%	0.9
NESMA procenjena	92%	10%	30%	0.9
FPA	95%	12%	20%	0.95

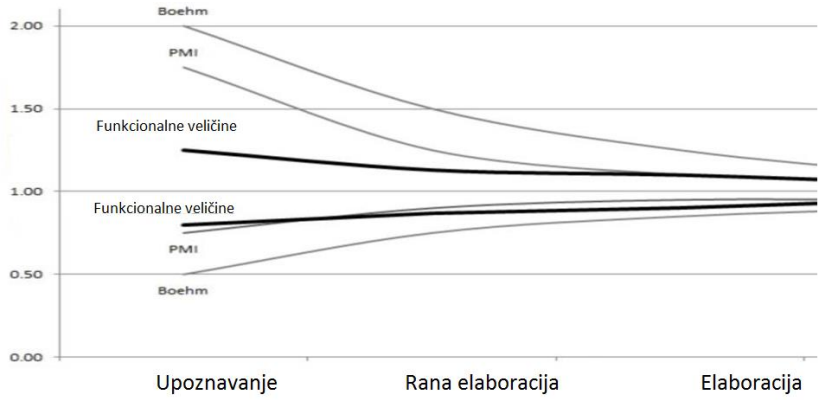
Mark II i NESMA procenjena metoda daju relativno slične rezultate uz srednju relativnu grešku od 10-13% i maksimalnu relativnu grešku od 30-65%. U poređenju sa dr Beomovim i PMI rezultatima ova tačnost je i više nego prihvatljiva.

Potpuna metoda funkcionalnih tačaka koja se koristi u kasnijim iteracijama daje sličnu pouzdanost kao i metode koje se koriste u ranijim iteracijama sa srednjom relativnom greškom od 12% i maksimalnom greškom od 20%.

Može se zaključiti da se u fazi elaboracije ne može dobiti bolji rezultat od greške od oko 10% bez obzira koja se metoda koristi. Ovaj procenat može biti sasvim prihvatljiv posebno ako se ima u vidu da postoje nefunkcionalni faktori kao što su performanse, lakoća upotrebe, interaktivnost, poznavanje tehnologije kao i produktivnost tima koji mogu uticati na funkcionalnu zavisnost veličine i napora koji se ne koriste u ovim tehnikama. Imajući ovo u vidu može se zaključiti da je tačnost procene napora koja se može dobiti tokom faze elaboracije korišćenjem isključivo funkcionalnih zahteva kreće od 10% do 13%.

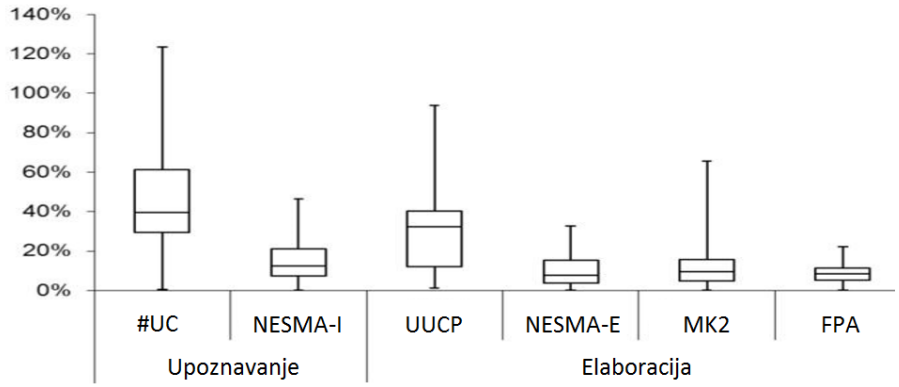
5.3. Zaključak

U ovom delu istraživanja je pokazano da metode procene napora zasnovane na funkcionalnim veličinama mogu da daju dobre rezultate procene. Na slici 33 je prikazana kupa nesigurnosti procena dobijena korišćenjem mernih metoda upoređena sa kupama nesigurnosti određenim na osnovu informacija PMI instituta i dr Boema.



Slika 33. Tačnost procena napora u odnosu na Boemove i PMI kriterijume.

Kao što se može primetiti greške procene su manje ili blizu očekivanih vrednosti. Na ovaj način se pokazuje da se metode procene funkcionalne veličine softvera mogu uspešno koristiti radi procene napora. Pored toga pokazano je da različite metode imaju različite tačnosti procene. Na slici 34 je prikazana raspodela grešaka procene napora na projektima u posmatranim metodama grupisanih po fazama projekta u kojima se mogu primeniti.



Slika 34. Opsezi grešaka procena napora različitim metodama.

Kao što se može primetiti na slici 34, NESMA indikativna metoda je odličan izbor za rane procene kada nema dovoljno informacija na raspolaganju. Greške inicijalne procene primenom ove metode se kreću u opsegu do 20%, sa maksimalnim vrednostima do 45%. NESMA procenjena metoda se pokazala kao izuzetno uspešna u ranim fazama elaboracije, dok FPA metoda daje odlične rezultate u kasnim fazama elaboracije.

6. Unapređenje metoda za procenu uticaja nefunkcionalnih zahteva

Pretpostavka koja će biti proverena u ovom delu istraživanja je da li se metode za procenu uticaja funkcionalnih i nefunkcionalnih zahteva mogu kombinovati. U skupu projekata koji se koriste u analizi su sakupljene informacije o funkcionalnim veličinama i određeni uticaji nefunkcionalnih zahteva primenom pravila iz FPA, UCP i COCOMO II metoda. Potom su se ovi uticaji kombinovali sa funkcionalnim veličinama određenim u prethodnom poglavlju kako bi se odredilo koji par pravila dovodi do najveće preciznosti procena napora.

6.1. Evaluacija metoda procene uticaja nefunkcionalnih zahteva

Pod pretpostavkom da su metode za procenu uticaja nefunkcionalnih zahteva nezavisne od metoda za procenu uticaja funkcionalnosti, može se postaviti pitanje zašto se ne bi kombinovala pravila za procene uticaja iz različitih metoda. Na ovaj način se mogu odabrati optimalna pravila za procenu.

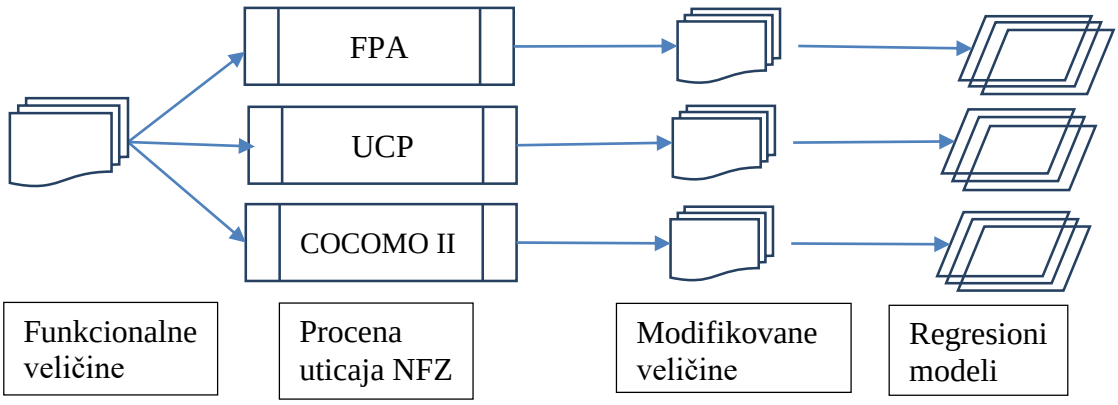
Kao polazna tačka istraživanja se koriste tačnosti procene napora na osnovu funkcionalnih veličina prikazane u prethodnom poglavlju. Tačnosti procena i parametri predikcionih modela su prikazani u tabeli 14.

Tabela 14. Tačnost procene predikcionih modela koji koriste funkcionalne veličine.

Veličina	MMRE	PRED(25)	R^2	p-value
NESMA-I	16%	0.8	0.88	1.31E-12
UUCP	31%	0.4	0.64	5.33E-07
NESMA-E	10%	0.9	0.95	9.89E-15
Mark II	13%	0.9	0.91	8.18E-13
UFP	11%	0.95	0.96	8.73E-16

Ove vrednosti se koriste kao polazna tačka za utvrđivanje da li se mogu primeniti različite metode procene uticaja nefunkcionalnih zahteva kako bi se poboljšale procene napora. Ideja koja će biti razrađena u ovom poglavlju je mogućnost primene parametara iz različitih metoda za procenu uticaja nefunkcionalnih zahteva, kako bi se dobile nove

vrednosti veličina sistema koje bi bile bolje korelisane sa vrednostima napora. Novi skupovi veličina se koriste u regresionim modelima kako bi se odredile greške procene na skupu projekata koji se analizira. Ovaj proces je prikazan na slici 35.



Slika 35. Primena različitih metoda procene NFZ uticaja na funkcionalne veličine.

Polazi se od funkcionalnih veličina projekata prikazanih u tabeli 14. Za svaki projekat se određuje uticaj nefunkcionalnih zahteva u skladu sa pravilima objašnjenim u sekcijama 4.3.1, 4.3.3, i 4.3.5. Uticaji nefunkcionalnih zahteva se primenjuju na funkcionalne veličine kako bi se dobile nove veličine projekata koje se koriste u regresionim modelima kojima se procenjuje napor na projektima.

U ovoj sekciji će biti primenjene najkorišćenije metode u praksi (tj. UCP, FPA i COCOMO II). Iako ostale metode imaju dobrih ideja za primenu pravila procene uticaja nefunkcionalnih zahteva one ipak nisu često korišćene tako da još nisu validirane u praksi na većem broju projekata.

6.1.1. Ocenjivanje NFZ uticaja UCP pravilima

Prva pretpostavka je da uticaj nefunkcionalnih zahteva određen pravilima definisanim u UCP metodi može da poboljša preciznost procena u svim metodama. Na svim projektima su određeni uticaji nefunkcionalnih parametara izraženi u vidu skupa EFC i TFC parametara, kao i vrednost uticaja nefunkcionalnih zahteva izražen u vidu VAF parametra. VAF vrednost je primenjena na svih pet funkcionalnih veličina po svakom projektu kako bi se dobila modifikovana veličina projekata korišćenjem formula (46).

$$Size = VAF * F\text{-}Size \quad (46)$$

U zavisnosti od metode kojom se određuje veličina softvera, funkcionalna veličina F-Size može biti UUCP, UFP, itd.

Standardna linearna regresija je primenjena na parovima veličina i napora po svim projektima sa novim veličinama i rezultati su upoređeni sa polaznim vrednostima iz tabele 14.

Tabela 15. Poboljšanje predikcionih modela UCP parametrima.

	Primenom F-Size veličina				Primenom veličine dobijene UCP metodom			
	MMRE	R^2	Pred(25)	p-value	MMRE	R^2	Pred(25)	p-value
NESMA-I	16%	0.81	0.88	1.31E-12	15%	0.85	0.91	7.55E-14
UUCP	31%	0.41	0.64	5.33E-07	27%	0.44	0.70	5.64E-08
NESMA-E	10%	0.91	0.95	9.89E-15	10%	0.96	0.95	4.97E-15
Mark II	13%	0.88	0.91	8.18E-13	13%	0.88	0.91	4.08E-13
UFP	11%	0.96	0.96	8.73E-16	10%	0.96	0.96	9.79E-17

U tabeli 15 se može primetiti da je uticaj nefunkcionalnih zahteva određen na osnovu UCP metode blago poboljšao procene u svim metodama.

Na ovaj način je pokazano da je pretpostavka da su metode procene uticaja funkcionalnih i nefunkcionalnih zahteva nezavisne i da isti metod za procenu uticaja nefunkcionalnih zahteva može da poboljša tačnost procena u svim metodama.

6.1.2. Ocenjivanje NFZ uticaja FPA pravilima

Isti eksperiment je ponovljen sa pravilima definisanim u FPA metodi. Uticaj nefunkcionalnih zahteva je određen na osnovu FPA pravila i svaka funkcionalna veličina u projektima je pomnožena vrednostima koje predstavljaju uticaj nefunkcionalnih zahteva korišćenjem formule (33). Linearna regresija je primenjena na polaznom skupu podataka sa ažuriranim veličinama i parametri su prikazani u tabeli 16.

Tabela 16. Poboljšanje predikcionih modela FPA parametrima.

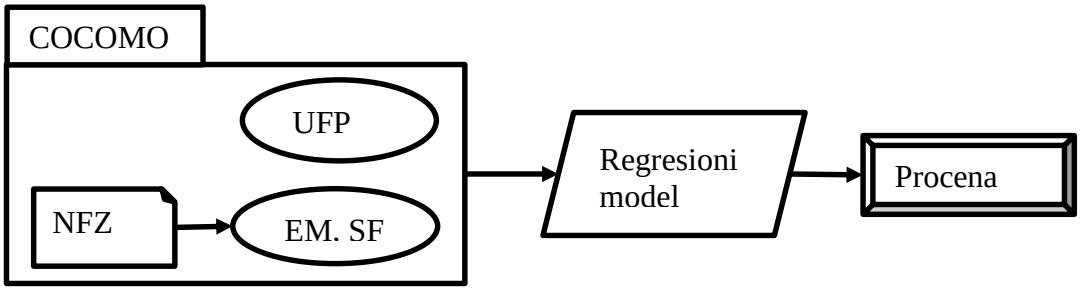
	Originalne F-Size veličine				Veličine ažurirane UCP metodom			
	MMRE	R^2	Pred(25)	p-value	MMRE	R^2	Pred(25)	p-value
NESMA-I	16%	0.81	0.88	1.3E-12	16%	0.77	0.89	5.07E-13
UUCP	31%	0.41	0.64	5.3E-07	31%	0.41	0.64	6.06E-07
NESMA-E	10%	0.91	0.95	9.9E-15	11%	0.91	0.94	1.52E-14
Mark II	13%	0.88	0.91	8.2E-13	13%	0.88	0.91	6.65E-13
UFP	11%	0.96	0.96	8.7E-16	10%	0.96	0.96	1.88E-16

U ovom slučaju se može primetiti da nije došlo do značajnijeg poboljšanja. Na posmatranom skupu podataka pravila za procenu uticaja UCP metodom daju bolje rezultate.

6.1.3. Primena COCOMO II parametara

COCOMO II je treći najčešće korišćeni formalni metod procene u praksi i najčešće korišćeni metod procene napora [Kassab *et al.*, 2014]. Ključna razlika između metoda procene veličine kao što su UCP i FPA i metoda procene napora kao što je COCOMO II je činjenica da je izlaz iz COCOMO II modela sama procena napora a ne neka veličina koja se dalje može koristiti radi procene.

COCOMO II model je kalibrisan na velikom broju projekata, ali to i dalje ne znači da će procenjena izlazna vrednost biti optimalna i za skup projekata posmatranih u radu. Zbog toga se u ovom radu vrednost koja se dobija iz COCOMO II modela ne usvaja direktno kao procena napora. Umesto toga, ona se posmatra kao apstraktna „veličina“ koja je bolje korelisana sa stvarnim naporom i koja može dovesti do tačnije procene napora. U ovom slučaju procenjujemo napor posredno, korišćenjem posebnog regresionog modela kojim se procenjuje napor na osnovu izlaza modela COCOMO II, kao što je prikazano na slici 36.



Slika 36. Primena linearne regresije na COCOMO II modelu.

U idealnom slučaju (ako COCOMO II model daje tačnu procenu napora), regresionim modelom bi se dobila funkcija sa linearnim koeficijentom 1. U tom slučaju bi ulaz i izlaz regresionog modela bili identični što bi značilo da originalna COCOMO II metoda idealno odgovara posmatranom skupu projekata.

Međutim, u ovom delu istraživanja se koristi izlaz COCOMO II modela kao ulazna promenljiva za regresionog modela. Na ovaj način, regresioni model može da poboljša procene u slučaju da originala metoda nije kalibrisana za posmatrani skup podataka.

Slično prethodnim slučajevima, funkcionalne veličine su modifikovane EM i SF parametrima, kako bi se dobila veličina na osnovu koje se procenjuje napor i kreira se regresioni model pomoću <veličina, napor> parova iz projekata. Parametri predikcionih modela ocenjeni pomoću LOOCV metode su prikazani su tabeli 17.

Tabela 17. Poboljšanje predikcionih modela COCOMO II parametrima.

	Primenom originalnih veličina				Modifikovanim veličinama			
	MMRE	R^2	Pred(25)	p-value	MMRE	PRED(25)	R^2	p-value
NESMA-I	16%	0.81	0.88	1.31E-1	8%	1.00	0.96	6.84E-2
UUCP	31%	0.41	0.64	5.33E-1	21%	0.63	0.75	4.33E-1
NESMA-E	10%	0.91	0.95	9.89E-2	8%	1.00	0.96	3.20E-2
Mark II	13%	0.88	0.91	8.18E-1	10%	0.96	0.95	1.36E-2
UFP	11%	0.96	0.96	8.73E-2	9%	1.00	0.96	4.90E-2

Bitno je primetiti da je tačnost predikcionih modela prikazanih u tabeli 17 bolja od tačnosti modela prikazanih u prethodnim sekcijama. Procena tačnosti po svim modelima osim UCP je manja ili jednaka 10% što je odličan rezultat. Iako je tačnost modifikovane

UCP metode lošija od drugih metoda, COCOMO II parametri su poboljšali procene sa 30% na 20%, što je opet veliki napredak.

6.2. Uproš̀avanje metoda procene uticaja nefunkcionalnih zahteva

U drugom delu istraživanja, proverava se da li se metode procene NFZ uticaja mogu prilagoditi i pojednostaviti na osnovu profila projekata. Pojednostavljeni postupci se mogu koristiti ranije u životnim ciklusima projekata kada nisu poznate sve informacije potrebne da se proceni NFZ uticaj. Ovo je važno zato što projektnim timovima trebaju procene što je pre moguće kako bi ih koristili za dalje odluke.

Pojednostavljenje pravila procene funkcionalnih zahteva je već istraženo u prethodnom poglavlju. NESMA indikativna i NESMA procenjena metoda su aproksimacije standardne FPA metode i daju dobre rezultate u ranim fazama projekta uprkos činjenici da ne uzimaju u obzir sve funkcionalne detalje. U ovom poglavlju će biti prikazano kako se mogu dodatno uprostiti i pravila za procenu uticaja nefunkcionalnih zahteva.

6.2.1. Aproksimacije pravila procene NFZ uticaja

Tokom analize vrednosti parametara koji opisuju uticaj nefunkcionalnih zahteva, može se primetiti veliki broj identičnih vrednosti po svim projektima. Identične vrednosti pojedinih parametara uzrokuju malu varijabilnost uticaja nefunkcionalnih zahteva.

Mala varijabilnost uticaja je ekvivalentna množenju svih funkcionalnih veličina istom konstantom što ne poboljšava značajno procene. Razlog za malu varijabilnost parametara leži u činjenici da su originalni parametri definisani i kalibrisani na velikom broju različitih projekata kako bi bili primenljivi u najširem mogućem broju projekata. Međutim, svaki skup projekata ima svoje specifičnosti koje opšti model ne može da uzme u obzir.

Skup projekata koji se analizira u ovom istraživanju se sastoji od veb aplikacija gde su ocene lakoće instalacije i mogućnosti prenošenja na druge platforme iste u svim projektima. Pored toga, svi projekti su nastali iz iste kompanije koja na svim projektima imala CMMI nivo zrelosti 2. Stoga je vrednost parametra zrelost procesa (PMAT) konstantna u svim projektima i ne uvodi nikakve dodatne informacije u procene modela.

Imajući ovo u vidu, može se zaključiti da neki parametri u postojećim metodama ne doprinose poboljšanju tačnosti procena napora.

Trenutno istraživači pokušavaju da smanje skup parametara za ocenu uticaja NFZ [Ochodek *et al.*, 2011] bez gubljenja preciznosti. U ovom radu je primenjen sličan princip.

U ovom delu analize se modifikuju COCOMO II pravila procene, ali ne i FPA / UCP pravila zato što je COCOMO II metoda identifikovana kao optimalna za procenu NFZ uticaja na posmatranom skupu podataka. Cilj istraživanja je dodatno uprostiti optimalnu metodu kako bi bila primenljiva što je moguće ranije. Zbog toga je identifikovan podskup COCOMO II parametara koji se mogu koristiti radi jednostavnije procene NFZ uticaja.

6.2.2. Odabir parametara

Kao kriterijum za izbor parametara koji će biti korišćeni radi procene NFZ uticaja koristi se standardna devijacija ocena parametara opisana u sekciji 2.4.4 [Merkle i Vasic, 1997]. Varijabilnost se definiše kao vrednost standardne devijacije ocena pojedinačnih vrednosti parametara po svim projektima. U slučaju da sve vrednosti nekog parametra imaju iste vrednosti u svim projektima, taj parametar ne donosi neku novu informaciju u predikcioni model. U tom slučaju standardna devijacija će biti nula. Za parametre čije se vrednosti značajnije razlikuju po projektima standardna devijacija će biti veća.

Koristeći standardnu devijaciju ocena parametara, identifikovan je skup parametara čije ocene imaju vrednosti standardne devijacije između 0,5 i 0,8 – kompleksnost (CPLEX), pouzdanost (RELY), iskustvo u domenu aplikacije (APEX), sposobnost analitičara (ACAP) i iskustvo u platformi (PLEX).

Ostali parametri ili slabo utiču na linearni koeficijent (sa standardnom devijacijom manjom od 0.2) ili skoro uopšte ne utiču na procenu usled relativno male standardne devijacije (manje od 0.1).

Pošto se odbace parametri sa malim standardnim devijacijama, na osnovu preostalih pet parametara se dobija formula (47) za procenu napora:

$$Effort = \prod_{i=1}^{5} EM_i * USize^{1.16} \tag{47}$$

U formuli se koristi pet EM parametara iz COCOMO II modela sa najvećom standardnom devijacijom. U posmatranom skupu projekata SF parametri imaju izuzetno malu standardnu devijaciju, tako da eksponent malo varira oko srednje vrednosti 1.16. U formuli se ignoriše i ta varijacija i koristi se prosečna vrednost eksponenta na svim projektima. Za razliku od standardnog COCOMO II modela, ne koristi se predefinisana konstanta 2.94 koja množi veličinu pošto će optimalni koeficijent biti dobijen linearnom regresijom.

6.2.3. Evaluacija

Kao i u prethodnim slučajevima, određene su nove vrednosti veličina projekata na osnovu originalnih funkcionalnih veličina i pet odabranih parametara iz COCOMO II modela, i primenjena je linearna regresija na parove veličina i napora po svim projektima u skupu. Pošto su određeni optimalni parametri regresionih modela, određene su i procene napora za svaki projekat na osnovu regresione formule.

Parametri modela regresije dobijenih posredstvom funkcionalnih veličina modifikovanih smanjenim skupom parametara su prikazani u tabeli 18.

Tabela 18. Poboljšanje predikcionih modela uprošćenim COOCMO II modelom.

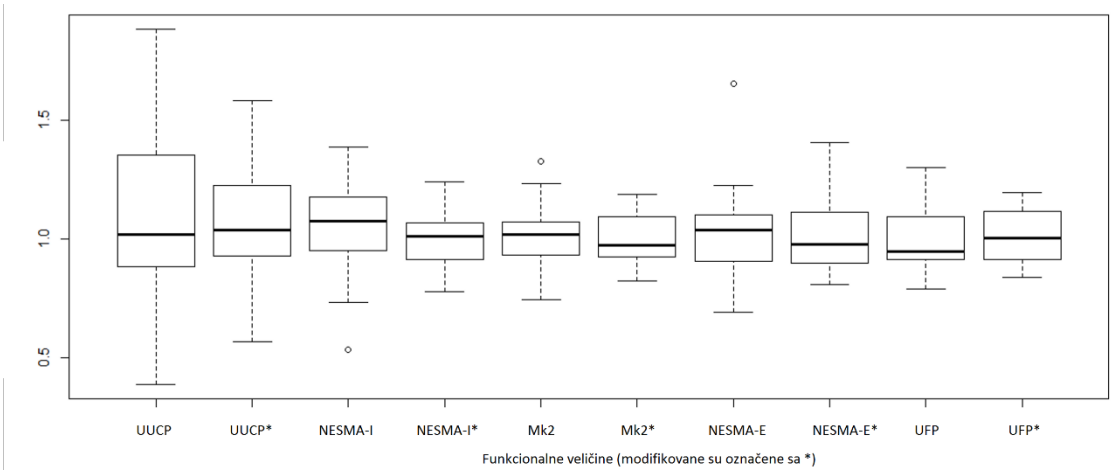
	Polazni modeli				Uprošćeni modeli			
Veličina	MMRE	PRED(25)	R^2	p-value	MMRE	R^2	Pred(25)	p-value
NESMA-I	16%	0.81	0.88	1.3E-12	10%	1.00	0.95	1.54E-17
UUCP	31%	0.41	0.64	5.3E-07	20%	0.67	0.77	1.78E-09
NESMA-E	10%	0.91	0.95	9.9E-15	9%	1.00	0.97	6.88E-17
Mark II	13%	0.88	0.91	8.2E-13	11%	0.96	0.95	1.66E-15
UFP	11%	0.96	0.96	8.7E-16	10%	1.00	0.95	4.22E-15

Može se primetiti da približne metode pružaju dobru tačnost procene – svi modeli sem modela koji koristi UCP kao osnovnu funkcionalnu veličinu imaju greške procene oko 10%, što je odličan rezultat. Kombinacija NESMA indikativne metode i odabranih parametara zahteva samo sedam parametara koji će se procenjivati (tj. broj internih i eksternih fajlova za NESMA indikativnu metodu, i pet NFZ parametara) sa greškom od oko 10%.

NESMA procenjena metoda, koja zahteva sedam parametara koji će se procenjivati (dva parametra za ukupan broj internih ili eksternih fajlova, i tri parametra za broj ulaza, izlaza

i upita) i pet odabranih NFZ parametara, daje još malo bolje tačnosti od originalne FPA metode koja zahteva 32 parametara (15 potrebnih radi procene UFP veličine i 17 potrebnih radi određivanja VAF faktora).

Na slici 37 je upoređena raspodela Z parametara modifikovanih metoda procene napora predstavljenih u ovoj sekciji i originalnih metoda predstavljenih u prethodnom poglavlju:



Slika 37. Poređenje raspodele Z-parametara predikcionih modela.

Kao što se može primetiti, uprošćena metoda smanjuje opseg grešaka. Ovi rezultati opravdavaju primenu i validnost predložene prilagođene metode. Izbor parametara najprimenljivijih za određeni skup podataka ili profil softverske organizacije može dovesti ne samo do jednostavnijih, nego i do preciznijih modela procene.

Pored poređenja sa polaznim funkcionalnim modelima, u tabeli 19 je prikazano poređenje uprošćenih modela i modela dobijenih primenom COCOMO II pravila za procenu uticaja nefunkcionalnih zahteva.

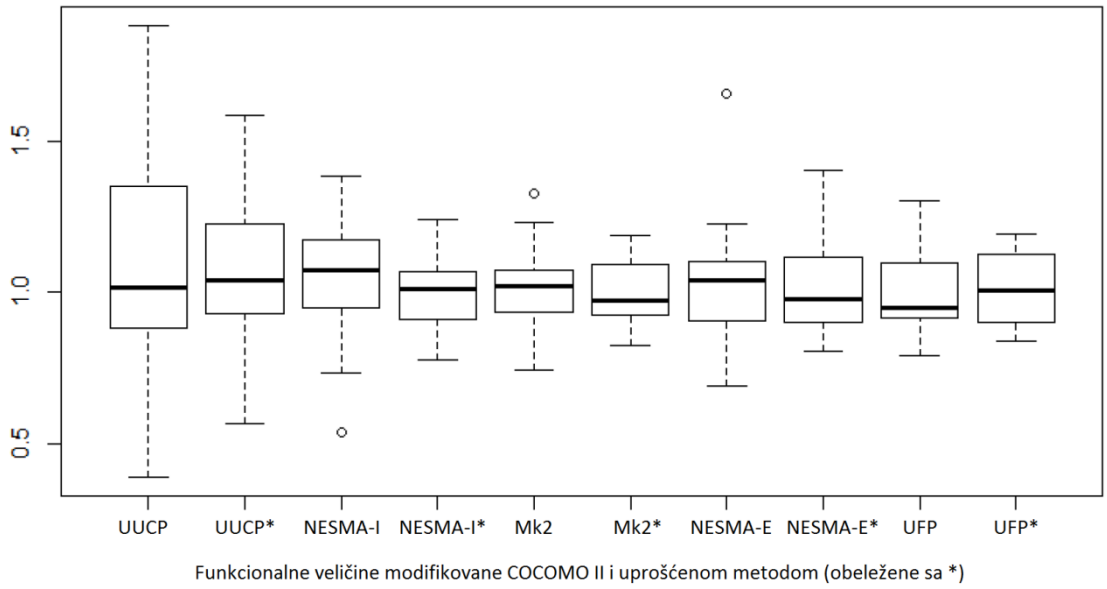
Tabela 19. Poređenje polaznog i uprošćenog COOCMO II modela.

	Polazni COCOMO II model				Uprošćeni modeli			
	MMRE	PRED(25)	R2	p-value	MMRE	R^2	Pred(25)	p-value
NESMA-I	8%	1.00	0.96	6.84E-18	10%	1.00	0.95	1.54E-17
UUCP	21%	0.63	0.75	4.33E-09	20%	0.67	0.77	1.78E-09
NESMA-E	8%	1.00	0.96	3.20E-16	9%	1.00	0.97	6.88E-17
Mark II	10%	0.96	0.95	1.36E-15	11%	0.96	0.95	1.66E-15
UFP	9%	1.00	0.96	4.90E-16	10%	1.00	0.95	4.22E-15

Uprošćeni model je bitno uporediti sa najoptimalnijim originalnim metodom kako bi se utvrdilo da li je došlo do poboljšanja rezultata. Pored toga, bitno je pokazati da li je uprošćenjem pravila COCOMO II došlo do smanjenja tačnosti i u kojoj meri.

Poređenjem rezultata može se primetiti da uprošćeni model ne unosi neku značajnu grešku u poređenju na COCOMO II model. Greške su veće za po 1-2% što je prihvatljivo, posebno imajući u vidu da uprošćeni model ima manje parametara koje treba oceniti i da se može koristiti ranije u životnom ciklusu projekta.

Na slici 38 se može videti poređenje Z odnosa dobijenih određivanjem uticaja COCOMO II metodom u odnosu na Z odnose dobijene podskupom COCOMO II parametara.



Slika 38. Poređenje COCOMO II modela sa uprošćenim modelom.

Može se primetiti da uprošćeni model smanjuje opseg grešaka po svim funkcionalnim veličinama što je još jedan dokaz validnosti inicijalne pretpostavke.

7. Metode procene napora projektnih zadataka

Većina metoda za procenu napora se fokusira na pokušaje procene napora na samom projektu. Međutim mali broj radova se bavi problemom procene napora pojedinih tipova projektnih zadataka. Trenutno nema dovoljno radova i istraživanja koje govore kako proceniti napor potreban za analizu zahteva, specifikaciju, testiranje, itd.

Parcijalna procena napora na projektu, tj. grupa projektnih zadataka, može biti korisna u sledećim slučajevima:

1. Softverske kompanije mogu dati kratkoročne procene napora koje bi obuhvatile samo analizu i specifikaciju kako bi klijenti znali koliko će početne faze projekta da koštaju,
2. Procene izvršene na osnovu veličina sistema se mogu primenjivati na projektne zadatke oko kojih eksperti ili projektni timovi ne mogu da postignu konsenzus o procenama napora,
3. Parcijalne procene se mogu kombinovati sa ekspertskim tako što eksperti procene one projekte zadatke za koje su sigurni a ostali se procene na osnovu veličine sistema,
4. Parcijalne procene napora na grupama od jednog ili više tipova projektnih zadataka se mogu koristiti i radi procene napora na projektu. U slučaju da se može proceniti odnos napora koji će biti utrošen na pojedinim projektnim zadacima u odnosu na napor potreban na projektu (npr. na osnovu istorijskih podataka ili iskustva), moguće je proceniti napor na projektu na osnovu napora procenjenih projektnih zadataka.

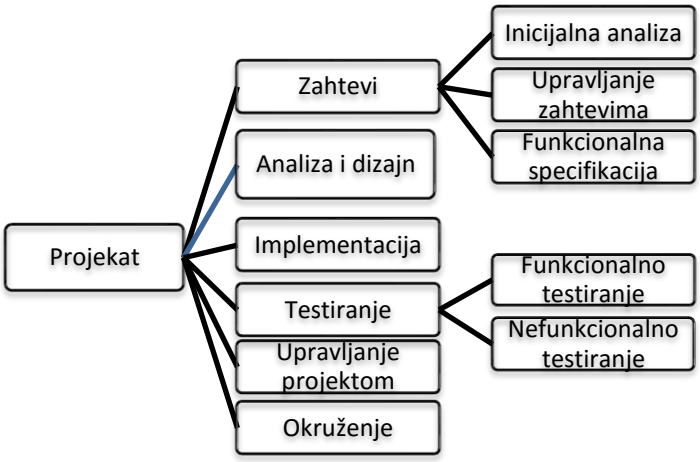
Problem procene pojedinih tipova projektnih zadataka nije temeljno istražen u literaturi. Vatson [Watson and McCabe, 1996] i Parvez [Parvez, 2013] su analizirali kako se ciklomatska kompleksnost i broj slučajeva korišćenja mogu koristiti radi procene napora potrebnog za testiranje softverskih aplikacija. Ovaj deo istraživanja je generalizacija te ideje. U ovom poglavlju će biti ispitano kako se veličina projekta može iskoristiti radi procene napora na projektnim zadacima.

U ovom poglavlju će biti ispitano koliko efikasno se mogu koristiti veličine projekata radi procene napora potrebnog da se završe određene vrste projektnih zadataka. Tokom analize će se koristiti UCP metoda kao najzastupljenija metoda za procenu veličine u praksi [Kassab *et al.*, 2014]. Hipoteza je da korelacija između UCP / UUCP veličina i napora potrebnog za završetak pojedinih projektnih zadataka nije uniforma. Dakle, mogu

se identifikovati projektni zadaci koji se mogu bolje proceniti na osnovu veličine softvera kao i oni koji degradiraju procene i predstavljaju slabosti procene pomoću UCP veličine.

7.1. Klasifikacija projektnih zadataka

Kao prvi korak analize izvršena je klasifikacija projektnih zadataka. U analizi je korišćena klasifikacija prikazana na slici 39 [Popovic *et al.*, 2015].



Slika 39. Klasifikacija projektnih zadataka.

Kategorije na prvom nivou su standardne discipline koje se mogu naći u većini modela životnog ciklusa softvera kao što je UP [Kruchten, 2000].

Neke kategorije su sumarne i obuhvataju više potkategorija, npr. pod kategorijom zahtevi se nalaze potkategorije „Inicijalna analiza“, „Upravljanje zahtevima“ i „Funkcionalna specifikacija“. Kategorija „Testiranje“ obuhvata dve potkategorije „Funkcionalno testiranje“ i „Nefunkcionalno testiranje“. Svaka kategorija je povezana na projektne zadatke u projektima.

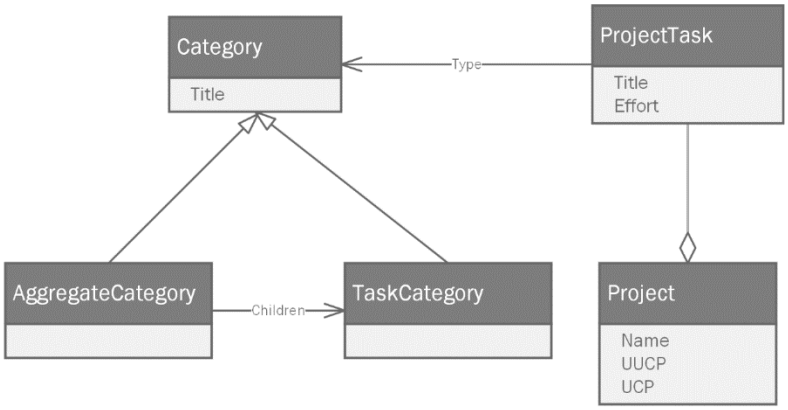
7.2. Analiza podataka

U ovoj sekciji će biti objašnjene strukture podataka, modeli i alati korišćeni u analizi. Analiza se svodi na kreiranje predikcionih modela kojima se na osnovu veličine projekata procenjuju napori potrebni po svakoj kategoriji projektnih zadataka prikazanoj na slici 39. Potom se porede parametri predikcionih modela po svakoj kategoriji kao i njihove

greške procene. Da bi se izvršila ovakva analiza potrebno je definisati odgovarajuće modele podataka kao i alate i jezike koji će se koristiti u analizi.

7.2.1. Model podataka

U analizi je korišćen model podataka prikazan na slici 40.

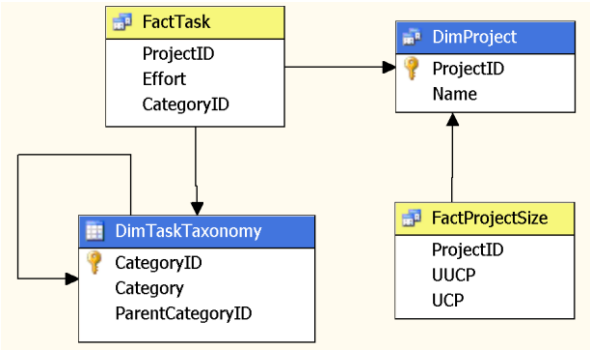


Slika 40. Model podataka korišćen u analizi.

Osnova za analizu su projektni zadaci koji pripadaju pojedinim projektima. Projektni zadaci su klasifikovani na osnovu kategorija koje mogu biti ili agregatne (tj. mogu obuhvatati druge kategorije) ili krajnje (*TaskCategory* na slici 40). Sumarne i krajnje kategorije korišćene u analizi su prikazane na slici 39.

7.2.2. Skladištenje podataka u OLAP kockama

U analizi se vrši poređenje predikcionih modela po različitim kategorijama. Radi ovakve analize potrebno je dizajnirati sistem koji omogućava da se skladište podaci o projektima i kategorijama i omogući poređenje predikcionih modela. U ovoj analizi je odabran multi-dimenzionalni OLAP model [MDX, 2012] prikazan na slici 41.

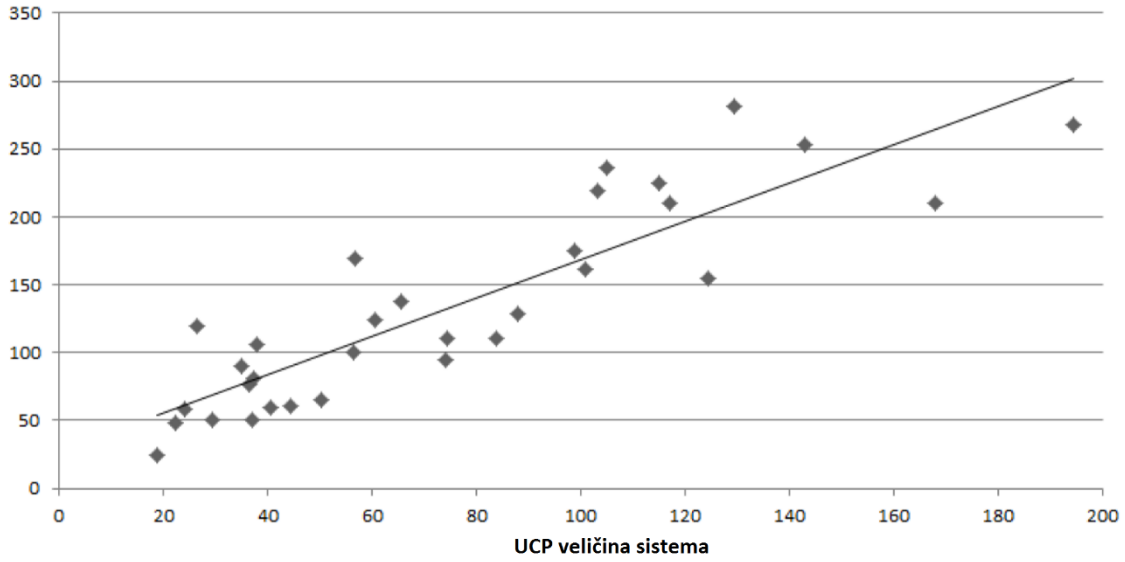


Slika 41. Multi-dimenzionalni OLAP model korišćen u analizi.

Multi-dimenzionalni model je usaglašen sa modelom podataka prikazanim na slici 40. U modelu se mogu primetiti dve dimenzije po kojima će se vršiti analiza – projekti (predstavljeni *DimProject* dimenzijom) i kategorije (predstavljene *DimTaskTaxonomy* dimenzijom). Podaci koji će se analizirati (ili činjenice u OLAP terminologiji) su predstavljene *FactTask* i *FactProjectSize* činjenicama.

7.2.3. Predikcioni modeli

Radi procene napora određenih projektnih zadataka koristi se linearna regresija. Na slici 42 je prikazan linearni model koji predstavlja zavisnost napora uloženog na projektu i veličine izražene u tačkama slučajeva korišćenja.



Slika 42. Procena napora pomoću tačaka slučajeva korišćenja.

Regresioni model prikazan na slici 42 se može koristiti radi procene napora pomoću UCP veličine. Tokom analize u ovom poglavlju će biti napravljeni regresioni modeli kojima se procenjuje napor na projektnim zadacima (testiranje, analiza, itd.) umesto napora projekta u celini.

Iako postoji dobra korelacija između veličine i napora na projektu, što znači da se napor na projektu može odrediti linearnom regresijom u zavisnosti od veličine, ne može se direktno zaključiti da će linearna regresija biti primenljiva i u slučaju funkcionalnih zavisnosti između napora na projektnim zadacima i veličine sistema. Da bi model linearne

regresije mogao da se koristi, potrebno je da prođe statističke testove opisane u sekciji 2.4.6:

1. H_{01}: Greške su normalno distribuirane, što se testira *Shapiro-Wilk* testom,
2. H_{02}: Greške su homoskedastične, što se testira *Breush-Pagan* testom,
3. H_{03}: Greške moraju da budu nezavisne, što se testira *Durbin-Watson* testom.

Svaki model linearne regresije koji ne prođe ove testove se odbacuje i zaključuje se da se on ne može koristiti radi procene napora.

7.2.4. Analiza OLAP podataka pomoću MDX upita

Predstavljanje podataka u obliku multi-dimenzionalne OLAP kocke omogućava da se izvrši analiza podataka pomoću multi-dimenzionalnih izraza (MDX – engl. Multi-dimensional expressions). Pored mogućnosti da se podaci agregiraju i presecaju po različitim dimenzijama, OLAP i MDX omogućavaju analizu velikim brojem ugrađenih funkcija kao što su *LinRegPoint*, koja određuje tačku na linearnoj regresionoj pravi na osnovu ulazne promenljive i skupa tačaka kojima se definiše regresioni model, ili *LinRegR2* koja direktno određuje koeficijent determinisanosti modela. Na slici 43 je prikazan MDX upit kojim se vrši analiza podataka u OLAP kocki.

```
WITH
MEMBER Estimate AS
        LinRegPoint(UCP, Projects, Effort, UCP)
MEMBER [R-Square]  AS
        LinRegR2(NONEMPTY(Projects, Effort), UCP, Effort)
MEMBER MMRE AS
        Avg(NONEMPTY(Projects, Effort), ABS(Estimate-Effort)/Effort)
MEMBER MMER AS
        Avg(NONEMPTY(Projects, Effort), ABS(Estimate-Effort)/Estimate)
SELECT        {[Task Types]} ON ROWS,
        {[R-Square], MMRE, MMER} ON COLUMNS
FROM MeasurementCube
```

Slika 43. Tačnost procene po projektnim zadacima.

Procena (engl. *Estimate*) se definiše kao ona tačka na linearnoj regresionoj pravi koja se dobijena na osnovu (UCP, Effort) parova po svim projektima u kocki. Kao nezavisna promenljiva za računanje procene se koristi četvrti parametar UCP koji predstavlja veličinu projekta izraženu u tačkama slučajeva korišćenja. Radi procene vrednosti se koristi standardna MDX funkcija *LinRegPoint*.

Koeficijent determinizma R^2 se računa na sličan način primenom postojeće *LinRegR2* funkcije. Greške procene modela MMRE i MMER se određuju na osnovu stvarnih i procenjenih vrednosti napora.

Kao rezultat upita se dobijaju tipovi projekata ([Task Types]) po redovima i R^2, MMRE, i MMER parametri regresionih modela po kolonama.

7.3. Evaluacija

Pošto je definisan model i proces analize, mogu se prikazati rezultati MDX upita. U tabeli 20 su prikazani parametri regresionih modela podeljeni po tipovima projektnih zadataka i veličinama koje se koriste u UCP metodi.

Tabela 20. Tačnosti procene tipova zadataka UUCP i UCP veličinama.

	UUCP				UCP			
Tip zadatka	R^2	MMRE	MMER	Pred(25)	R^2	MMRE	MMER	Pred(25)
Zahtevi	0.66	25%	22%	63%	0.68	22%	21%	69%
Implementacija	0.63	28%	26%	47%	0.70	24%	22%	56%
Testiranje	0.60	34%	31%	44%	0.64	30%	28%	50%
Definisanje opsega posla	0.62	20%	19%	59%	0.66	18%	18%	66%
Funkcionalna specifikacija	0.79	17%	17%	81%	0.78	17%	17%	81%
Funkcionalno testiranje	0.83	16%	15%	59%	0.82	16%	15%	56%
Projekat	0.70	31%	28%	47%	0.78	26%	24%	56%

U tabeli nisu prikazane kategorije koje nisu prošle statističke testove linearne regresije. Model linearne regresije ne može da se primeni na projektne zadatke u ovim kategorijama tako da one negativno utiču na procenu napora samog projekta pošto je očigledna nelinearna zavisnost između njih i veličine projekta. Što se tiče ostalih kategorija, može se primetiti da je polazna pretpostavka tačna. Greške procene se razlikuju među kategorijama i mogu se podeliti na:

1. Male greške koje se dobijaju u procenama tipova projektnih zadataka koji su dobro korelisani sa vrednostima UCP veličina softvera. U ovu grupu spadaju definisanje opsega posla, funkcionalno testiranje i funkcionalna specifikacija.

2. Greške procene projektnih zadataka čije su vrednosti blizu greškama procena napora projekta. U ovu grupu spadaju najveći zadaci u projektima kao što su zahtevi, implementacija i testiranje. Ovi projektni zadaci, koji predstavljaju većinske delove projekata, određuju preciznost UCP metode.
3. Greške procene ostalih tipova projektnih zadataka su dosta veće od grešaka procene napora na projektu. Ovi tipovi projektnih zadataka (npr. upravljanje projektom ili definisanje okruženja) degradiraju procene napora na projektu usled slabe korelacije sa UCP veličinama.

Ovi rezultati pokazuju da je inicijalna hipoteza bila tačna. Mogu se identifikovati tipovi projektnih zadataka koji su bolje korelisani sa veličinama softvera od drugih projektnih zadataka.

7.4. Validacija rezultata

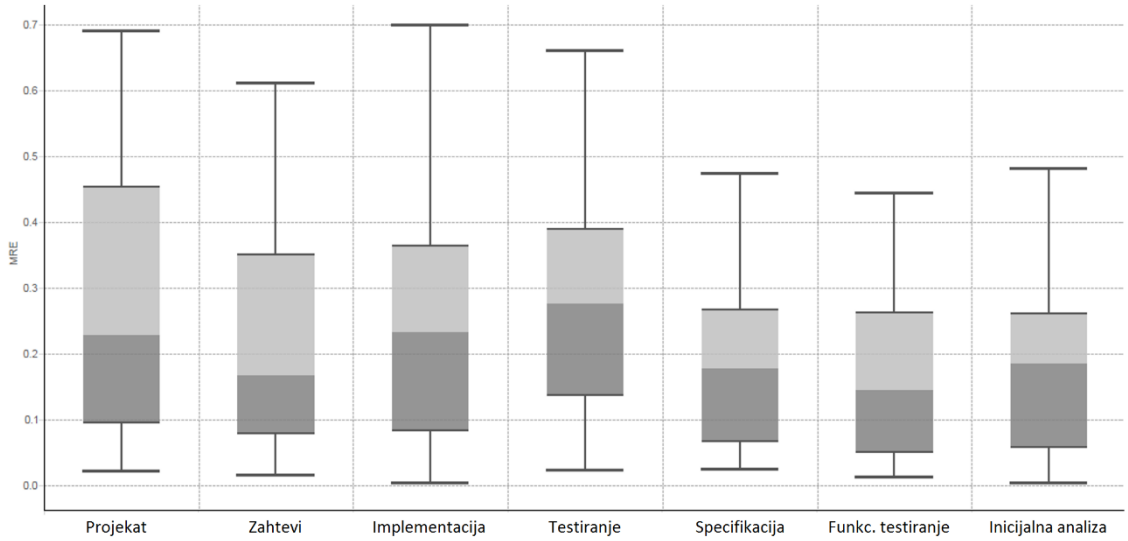
Da bi se validirali rezultati predstavljeni u prethodnoj sekciji potrebno je izvršiti unakrsnu validaciju rezultata. U ovoj sekciji će biti primenjena unakrsna LOOCV validacija opisana u sekciji 2.4.5. Rezultati unakrsne validacije se mogu dobiti korišćenjem MDX upita prikazanog na slici 44.

```
WITH
MEMBER [Project Size] AS UCP
MEMBER Estimate AS
        LinRegPoint(
                [Project Size],
                Except(Projects, [Dim Project].[Project].CurrentMember),
                Effort, [Project Size])
MEMBER MRE AS ABS(Estimate-Effort)/Effort
MEMBER Z AS Estimate/Effort
SELECT
        [Task Types]*[Dim Project].[Project].CurrentMember ON ROWS,
        MRE ON COLUMNS
FROM UCPCube
```

Slika 44. Validacija rezultata procene pomoću LOOCV metode.

Upit je sličan MDX upitu prikazanom na slici 43 kojim se procenjuje napor uz razliku da je trećem parametru *LinRegPoint* funkcije prosleđen izraz koji iz skupa projekata nad kojima se vrši linearna regresija izbacuje trenutni projekat, što je u skladu sa procesom LOOCV validacije. MDX upit vraća relativne greške procene i Z faktore za svaki projekat koji se koristi u validacionom skupu.

Na slici 45 je prikazana distribucija Z faktora po tipovima projektnih zadataka.



Slika 45. Raspodela grešaka procene po tipovima projektnih zadataka.

Neki tipovi projektnih zadataka su vrlo predvidljivi i postoji jaka korelacija između napora potrebnih da se implementiraju i veličina projekata. Korišćenje istorijskih podataka i informacija o produktivnosti tima dovodi do procena visoke tačnosti za ove grupe projektnih zadataka – većih od tačnosti procene napora na projektima.

Međutim, procene napora potrebnih za konfigurisanje, upravljanje projektom, analizu i dizajn dosta odstupaju od stvarnih vrednosti tako da se ne mogu proceniti na ovaj način.

7.5. Zaključak

U ovom poglavlju je pokazano da se mogu identifikovati projektni zadaci koji su bolje korelisani sa UCP veličinama projekta od ostalih projektnih zadataka. Dobra korelacija između projektnih zadataka i UCP veličine omogućava da se napor potreban da se identifikovani projektni zadaci implementiraju tačnije proceni upotrebom UCP veličine i linearne regresije.

Ovaj rezultat je bitan zato što omogućava unapređenje procena napora kombinovanjem formalnih i subjektivnih metoda kao što će biti pokazano u sledećem poglavlju.

8. Kombinovanje formalnih i subjektivnih metoda procene napora

U prethodnim poglavljima fokus istraživanja je bio isključivo na parametarskim metodama kojima se na osnovu veličina sistema procenjuje napor potreban za implementaciju projekata ili projektnih zadataka.

Međutim, u procesu procene napora se ne mogu zanemariti i ostale metode kao što su ekspertske procene [Jorgensen, 2010].

I subjektivne i formalne metode procene napora imaju svoje prednosti i mane [Jorgensen *et al.*, 2009], tako da se ne može očekivati da će bilo koji pristup preovladati. Zato projektni timovi koriste metode procene koje najbolje odgovaraju njihovim potrebama. Ne može se osporiti Jorgensenova izjava da "ne postoji najbolja metoda ili model procene napora" [Jorgensen, 2014], tako da je potrebno ispitati kako se postojeće metode mogu kombinovati, iskoristiti prednosti pojedinih metoda, identifikovati i ublažiti slabosti koje u određenim metodama identifikuju.

Kao osnova za istraživanje u ovoj oblasti se koriste ideje predstavljene u prethodnom poglavlju tj. podela projekta na tipične grupe projektnih zadataka. Projekti su podeljeni po tipovima projektnih zadataka i za svaki tip su određene procene napora korišćenjem UCP veličine i na osnovu ekspertskih procena. U ovom poglavlju će biti predstavljeno kako se ove metode mogu kombinovati kako bi se dobile bolje procene napora projekta.

8.1. Poređenje formalnih i subjektivnih metoda

Kao prvi korak analize, izvršeno je poređenje grešaka procene napora dobijenih na osnovu UCP veličine i ekspertskih procena. Rezultati su prikazani u tabeli 21.

Tabela 21. Poređenje grešaka procene napora UCP metodom i procenom eksperta.

Parametar	UCP metoda	Ekspertska procena
MMRE	23%	28%
PRED(25)	62%	51%
R^2	76%	69%
Min-Max	0% - 35%	0% - 66%

Vrednosti parametara dobijenih UCP metodom su već predstavljeni u prethodnom poglavlju. Parametri metoda ekspertske procene su dobijeni tako što su procene napora na projektima upoređene sa stvarnim vrednostima napora utrošenog na projektima.

Rezultati procene napora na osnovu UCP veličine i ekspertske procene su prihvatljivi i po Boemovim i po PMI rezultatima. Čak 90% procena je prihvatljivo po Boemovim očekivanjima za procene tokom definisanja zahteva, dok je 82% projekata prihvatljivo po PMI kriterijumu za inicijalne procene. Ekspertske procene imaju slične rezultate; 80% procena je prihvatljivo po Boemovim, dok je 76% procena prihvatljivo po PMI očekivanjima.

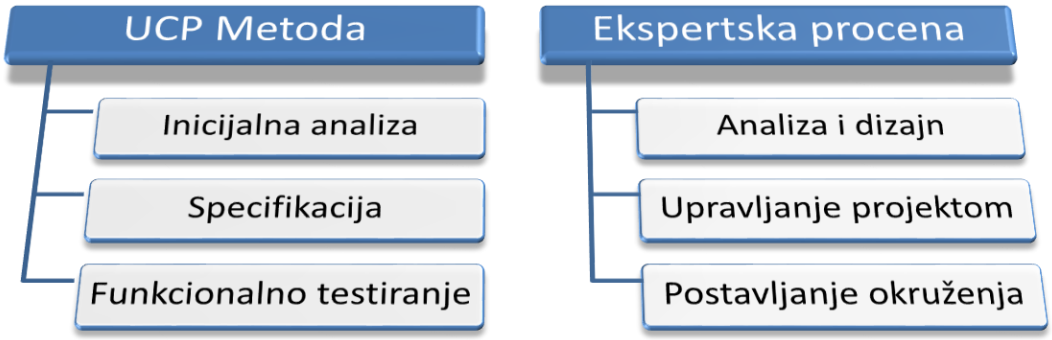
Pored poređenja grešaka procene napora na projektima, izvršeno je poređenje grešaka koje se dobijaju kada se procene pojedini projektni zadaci UCP metodom ili ekspertskom procenom. Rezultati poređenja su prikazani u tabeli 22.

Tabela 22. Greške procene napora po tipova projektnih zadataka.

	UCP			Procena eksperta	
	MMRE	PRED(25)	R^2	MMRE	PRED(25)
Inicijalna analiza	16%	82%	82%	25%	56%
Funkcionalna specifikacija	17%	74%	85%	22%	56%
Analiza i dizajn	64%	36%	22%	45%	46%
Razvoj	27%	49%	69%	37%	26%
Funkcionalno testiranje	16%	77%	88%	38%	56%
Ne-funkcionalno testiranje	59%	23%	23%	50%	36%
Upravljanje projektom	32%	41%	63%	13%	82%
Upravljanje zahtevima	50%	36%	13%	42%	23%
Okruženje	62%	28%	14%	16%	59%

U tabeli se mogu primetiti projektnih zadaci koji se tačnije mogu odrediti formalnim metodama (tj. UCP veličinom) i oni koji se tačnije mogu odrediti ekspertskom procenom.

Tipovi projektnih zadataka koji se bolje mogu proceniti na osnovu UCP veličine ili ekspertske procene su prikazani na slici 46.



Slika 46. Podela projektnih zadataka na osnovu optimalnih metoda procene.

Inicijalna analiza, specifikacija i funkcionalno testiranje imaju manje greške procene kada se procenjuju UCP metodom nego kada se ih procenjuje ekspert. Ovi tipovi projektnih zadataka su vrlo predvidljivi i postoji jaka korelacija između napora potrebnih da se implementiraju i veličina projekata. Korišćenje istorijskih podataka i informacija o produktivnosti tima za ove grupe projektnih zadataka dovodi do procena visoke tačnosti – većih od tačnosti procene napora na projektima.

Međutim, nefunkcionalno testiranje, postavljanje okruženja i upravljanje projektom se efikasnije procenjuju ekspertskim procenama. Ovi projektni zadaci se preciznije procenjuju heuristikama i iskustvom.

8.2. Kombinovanje ekspertskih i formalnih metoda

S obzirom da postoji razlika u tačnosti procene projektnih zadataka u zavisnosti od primenjene metode, ta činjenica je iskorišćena za unapređenje metoda procene napora. U ovoj sekciji će biti prikazano kako se UCP metoda i ekspertska procena mogu kombinovati u cilju dobijanja boljih procena napora na projektima.

8.2.1. Metoda sinteze

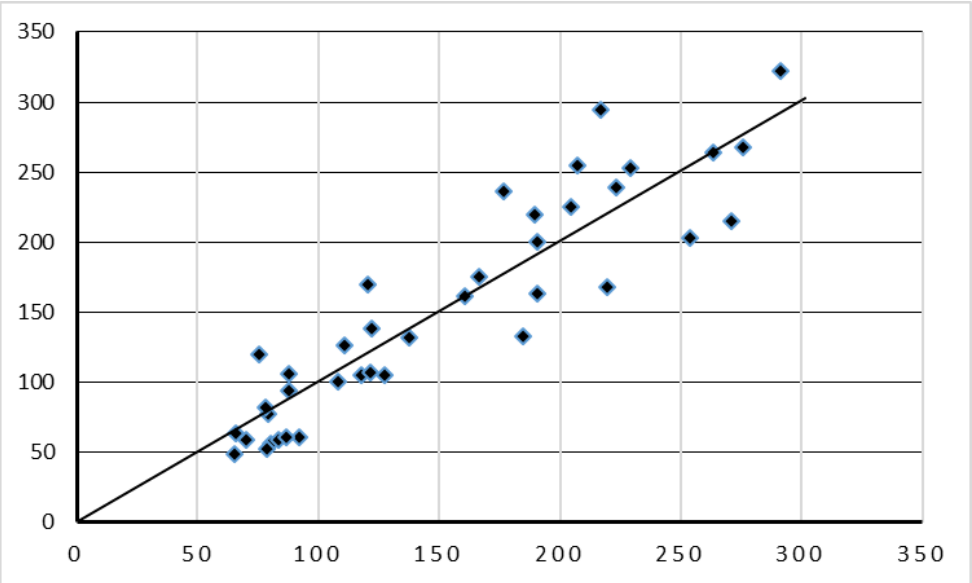
U uvodu je već opisana metoda sinteze kojom se procenjuju elementarni zadaci na projektu i sabiraju njihove procene napora. Ova metoda se odlično uklapa u podelu

projekta na vrste projektnih zadataka pošto se napor na projektu može sintetizovati na osnovu pojedinačnih procena tipova projektnih zadataka.

U ovom delu istraživanja koristi se sledeće pravilo – svaki tip projektnih zadataka se procenjuje onom metodom koja se pokazala kao primenljivija. Procena napora na projektu se dobija sintezom optimalnih procena delova projekta (tj. projektnih zadataka). Podela projektnih zadataka koji će se proceniti na osnovu UCP veličine ili ekspertske procene se vrši na osnovu greške procene, tj. MMRE vrednosti iz tabele 22.

1. Na osnovu ekspertske procene se procenjuju analiza i dizajn, nefunkcionalno testiranje, upravljanje projektom i upravljanje zahtevima.
2. Na osnovu UCP veličine se procenjuju inicijalna analiza, funkcionalna specifikacija, razvoj i funkcionalno testiranje.

Procena napora na projektu se dobija sabiranjem procena po projektnim zadacima. Na slici 47 su prikazani parovi procenjenih i stvarnih vrednosti napora po svim projektima.



Slika 47. Prikaz parova procenjenih i stvarnih vrednosti napora.

U idealnom slučaju parovi procenjenih i stvarnih vrednosti bi bili kolinearni i nalazili bi se na linearnoj pravoj prikazanoj na slici 47. Međutim, i ovi parovi su dobro korelisani uz koeficijent korelacije od 92%.

Poređenjem procenjenih i stvarnih vrednosti dobijaju se parametri procene prikazani u sledećoj tabeli.

Tabela 23. Tačnost metoda sinteze.

Parametar	Metoda sinteze	UCP metoda	Ekspertska procena
MMRE	19%	23%	28%
PRED(25)	64%	62%	51%
Min-Max (MRE)	0% - 52%	0% - 35%	0% - 66%

Metodom sinteze se dobija greška procene projekta od 19% uz PRED(25) = 64%. Može se primetiti da su parametri predikcionih modela dobijenih metodom sinteze bolji od originalnih metoda linearne regresije na osnovu UCP veličine, kao i ekspertskih procena. Na ovaj način se dobija greška procene bolja i od UCP metode i od ekspertske procene. Čak 90% projekata ima greške procene u okvirima očekivanih vrednosti po Boemovim i PMI kriterijumima.

8.2.1.1. Poboljšanje indirektne metode linearnom regesijom

Na slici 47 se može primetiti da su procene napora E^* koje su dobijene metodom sinteze dobro korelisane sa naporom. Na osnovu ove činjenice se može zaključiti da bi model linearne regresije definisan nad ovim parovima mogao da obezbedi bolje procene.

Model linearne regresije definisan nad ovim parovima je prikazan formulom (48):

$$Effort = 1.03 * E^* - 4.8 \quad (48)$$

Međutim, s obzirom da su koeficijenti linearne regresije 1.03 i -4.8 približno jednaki idealnim vrednostima (1, 0), na posmatranom skupu podataka ovaj model ne donosi značajna poboljšanja u poređenju sa prethodnom metodom. MMRE je manji za 1%, PRED(24) za 2%, dok je maksimalna greška manja za 4%.

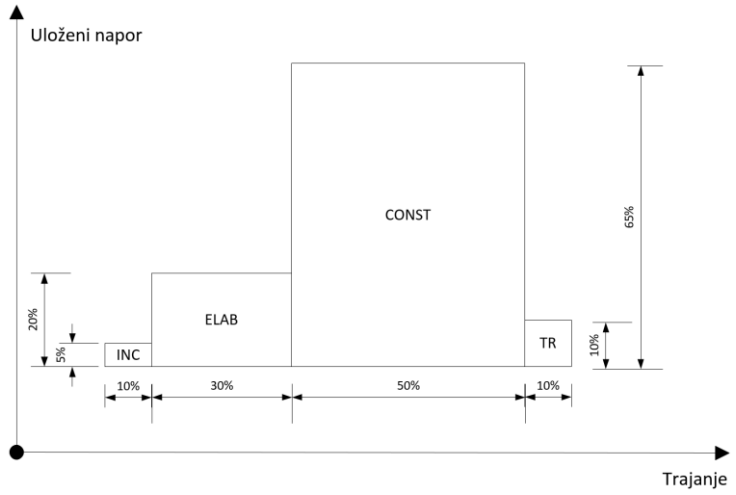
8.2.2. Indirektna metoda

U metodi sinteze su sabrani svi projektni zadaci kako bi se procenio napor potreban da se završi projekat. Međutim, može se primetiti da neke vrste projektnih zadataka ne mogu biti uspešno procenjene ni pomoću UCP veličine, niti ekspertskom procenom. Na primer, greške procene dobijene za analizu i dizajn, upravljanje zahtevima, i nefunkcionalno testiranje su i dalje visoke (greška procene je veća od 40% na osnovu obe metode). Loše procene ovih projektnih zadataka unose šum u procenu napora na projektu.

U cilju prevazilaženja ove slabosti, istraženo je da li postoji neki drugi način da se proceni napor projekta indirektno bez uključivanja ovih tipova zadataka.

8.2.2.1. Hipoteza

Pod pretpostavkom da procesi razvoja softvera imaju sličnu raspodelu napora uloženih u različitim fazama ili disciplinama projekta, može se pretpostaviti da je vreme potrebno da se završi projekat srazmerno vremenu potrebnom da se završe određene grupe projektnih zadataka. Iako se odnos napora potrebnog radi implementacije tipova projektnih zadataka i napora potrebnog da se implementira projekat može razlikovati, u literaturi se mogu naći izveštaji koji govore o očekivanim ili prosečnim raspodelama projektnih zadataka po disciplinama (npr. Inception – IN, Elaboration – ELA, Construction – CONST, Transition – TR koji se koriste u UP metodologiji) kao što je prikazano na slici 48 [Krutchen04].



Slika 48. Raspodela napora po disciplinama u UP metodologiji.

Iako trajanje i napor koji je potrebno uložiti u pojedine discipline može da se razlikuje od projekta do projekta, može se pretpostaviti da će se na konstrukciju (CONST) u proseku trošiti 65% napora projekta i da će trajati oko 50% vremena projekta. Ako bi se napor potreban da se završi konstrukcija precizno procenio nekim metodama, postoji mogućnost da bi se napor na projektu mogao proceniti primenom direktne proporcionalnosti. Isti princip bi se primenio i na druge discipline ili grupe disciplina.

Za razliku od metode sinteze, u kojoj se sabiraju sve procene projektnih zadataka, ideja indirektne metode koja će biti predstavljena u ovoj sekciji je da se procene samo oni

projektni zadaci koji imaju visoke preciznosti procene bilo UCP ili ekspertskom metodom. Ove procene se sabiraju kako bi se dobila veoma precizna procena dela projekta, i ta parcijalna procena se koristi za procenu napora na projektu. Pod pretpostavkom da je napor potreban da se implementira grupa projektnih zadataka srazmeran naporu potrebnom da se implementira projekat, može se pretpostaviti da će linearna regresija među ovim veličinama dati dobre procene.

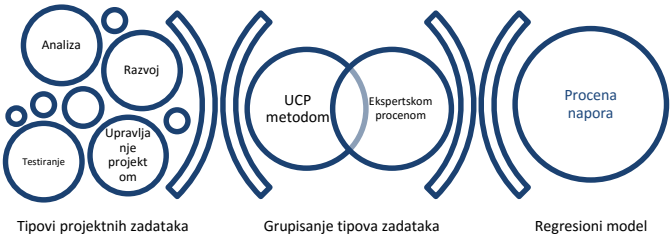
U analizi se ne koriste odnosi prikazani na slici 48. Discipline prikazane na slici 48 ne odgovaraju po nivou granularnosti projektnim zadacima koji su identifikovani u skupu projekata koji se koristi u analizi. Pored toga, ne može se dokazati da odnosi prikazani na slici 48 odgovaraju odnosima u posmatranom skupu projekata. Zbog toga se odnosi određuju na osnovu informacija predstavljenih u prethodnom poglavlju gde je već za svaki tip projektnih zadataka određen potreban napor. Ovi napori se koriste radi određivanja distribucije napora po projektnim zadacima čime se dobijaju odnosi koji su kalibrisani i optimizovani na konkretnom skupu podataka.

8.2.2.2. Analiza

Iz skupa projektnih zadataka su odabrani oni koji se mogu proceniti bar jednom metodom sa visokom preciznošću procene (tj. iz grupe tipova projektnih zadataka sa greškama između 13% i 17%). Na ovaj način su formirane dve grupe projektnih zadataka:

1. Grupa projektnih zadataka čiji se napor može precizno odrediti UCP metodom koju čine inicijalna analiza, funkcionalna specifikacija i funkcionalno testiranje.
2. Grupa projektnih zadataka čiji se ukupan napor može precizno odrediti ekspertskom procenom koju čine upravljanje projektom i postavljanje okruženja.

Određene su procene napora u ovim grupama i one su sabrane kako bi se dobila precizna procena grupe projektnih zadataka. Ova parcijalna procena je dalje korišćena kao ulaz u modelu linearne regresije na osnovu koga je procenjen napor projekta, kao što je prikazano na slici 49.



Slika 49. Podela tipova projektnih zadataka po optimalnim metodama procene.

Inicijalna analiza, funkcionalna specifikacija i funkcionalno testiranje su procenjeni na osnovu UCP veličine i linearne regresije, dok su za upravljanje projektom i postavljanje okruženja korišćene ekspertske procene. Zbir procena ovih projektnih zadataka predstavlja delimičnu procenu napora projekta – u daljem tekstu označena sa E_p^*, koja se računa po formuli (49).

$$E_p^* = \sum E_{tt}, MMRE(E_{tt}) < 0.2 \tag{49}$$

U formuli 49, E_{tt} je procena napora pojedinih projektnih zadataka. Procene se uključuju u parcijalnu sumu samo ako je odgovarajuća greška procene manja od 20%.

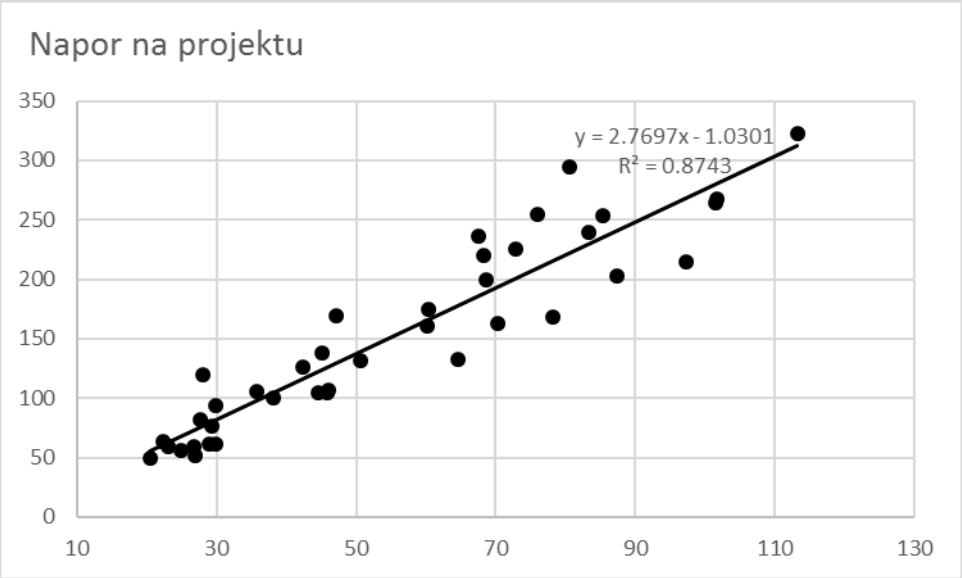
S obzirom da je pretpostavka da je napor potreban da se implementira projekat srazmeran vrednosti E_p^*, primenjuje se linearna regresija kojom se ispituje da li postoji linearna zavisnost između E_p^* i napora na projektu.

Model linearne regresije definisan nad ovim parovima je prikazan formulom (50):

$$Effort = 2.77 * E_p^* - 1.03 \tag{50}$$

Statistički dijagnostički testovi koji su opisani u sekciji 2.4.6. nisu odbacili pretpostavku da se linearna regresija ne može primeniti tako da je definisan linearni regresioni model između E_p^* i *Effort* veličina.

Linearna zavisnost napora na projektu i parcijalne procene grupe optimalnih projektnih zadataka je prikazana na slici 50.



Slika 50. Napor na projektu u zavisnosti od procene dela projektnih zadataka.

Kao što se može primetiti na slici, korelacija između nove promenljive E_p^* na osnovu koje se vrši procena i napora je bolja nego u prethodnim modelima. U dobijenom modelu R^2 koeficijent ima vrednost 0.87 što je odličan rezultat. Parametri dobijenog modela linearne regresije su prikazani u tabeli 24.

Tabela 24. Tačnost predikcionog modela dobijenog indirektnom metodom.

Parametar	Indirektna metoda	UCP metoda	Ekspertska procena
MMRE	15%	23%	28%
PRED(25)	85%	62%	51%
R^2	87%	76%	69%
Min-Max(MMRE)	3% - 42%	0% - 35%	0% - 66%

Linearna regresija daje grešku (MMRE) 15% i PRED(25) 85%. Greške su manje nego kod metode sinteze predstavljene u prethodnoj sekciji tako da se može zaključiti da indirektna metoda daje bolju tačnost procene. U 97% projekata, procene greške su u potpunosti usklađene sa PMI i Boemovim očekivanjima za rane procene, a 64% projekata ima greške koje su u granicama očekivanja po PMI budžetskim procenama.

8.2.2.3. Zaključak

U ovoj sekciji je predstavljena indirektna metoda za procenu napora u kojoj se napor na projektu procenjuje na osnovu onih projektnih zadataka koji se najpreciznije mogu proceniti bilo formalnim metodama bilo subjektivnom procenom.

U ovom slučaju koriste se samo oni tipovi zadataka koji mogu da se procene sa visokom preciznošću, a zatim se korišćenjem ove pouzdanije mere indirektno procenjuje napor. Rezultati pokazuju da ovaj pristup omogućava da se izbegne šum koji projektni zadaci, koji su loše korelisani sa naporom na projektu, unose u metodu sinteze.

8.2.3. Primena višestruke linearne regresije

U indirektnoj metodi koja je predstavljena u prethodnoj sekciji je definisana nova promenljiva E_p^* kao direktna suma parcijalnih procena tipova projektnih zadataka koja se koristi kao nezavisna promenljiva u regresionom modelu. Međutim, zbir procena najboljih tipova zadataka možda i nije najbolji način za novu meru.

Na primer, može se pretpostaviti da neki tipovi projektnih zadataka bolje utiču na procene, tako da se procenama zadataka mogu dodeliti težinski koeficijenti kako bi se dala prednost onim projektnim zadacima koji bolje utiču na procenu napora. U tom slučaju formula za određivanje parcijalne procene je prikazana u formuli (50).

$$E_p^* = \sum w_i * E_{tt}, MMRE(E_{tt}) < 0.2 \quad (50)$$

Može se primetiti da je formula (48) specijalni slučaj formule (50), gde su svi težinski koeficijenti 1. Ovakva veličina može dati bolje procene od običnog zbira projektnih zadataka. S obzirom da su svi koeficijenti u formuli (50) linearni, može se koristiti višestruka linearna regresija kako bi se oni odredili.

8.2.3.1. Hipoteza

Linearna kombinacija parcijalnih procena tipova zadataka, prikazanih u jednačini (50), trebalo bi da omogući bolje procene napora projekta. Pronalaženje optimalnih težinskih faktora može da smanji grešku procene napora na projektu. Ako su svi težinski koeficijenti linearne regresije 1, model će biti degradiran sa linearne kombinacije na prost

zbir, što je ekvivalentno prethodnom slučaju. Međutim, optimalni koeficijenti bi trebalo da omoguće bolje procene tačnosti.

8.2.3.2. Analiza

Linearna kombinacija procena tipova zadataka se može odrediti korišćenjem višestruke linearne regresije. U ovom slučaju, težinski koeficijenti predstavljaju linearne koeficijente višestruke linearne regresije. Radi određivanja funkcionalne zavisnosti i koeficijenata koristi se funkcija za određivanje linearnog modela u standardnom R paketu:

```
lm.part <- lm (effort ~ ucp.scope + ucp.spec + ucp.ftest + ex.pm + ex.env, ds)
```

Promenljiva *effort* predstavlja ciljnu vrednost koju treba proceniti. Vrednosti ucp.scope, ucp.spec i ucp.ftest su procenjeni napori za definisanje opsega (inicijalnu analizu), specifikaciju i funkcionalno testiranje dobijeni pomoću UCP veličine, dok su ex.pm i ex.env procene potrebne za upravljanje projektom i uspostavljanje okruženja dobijene ekspertskom procenom.

Model višestruke linearne regresije definisan nad ovim dimenzijama je prikazan formulom (51):

$$Effort = 1.59 * E_{scope}^{ucp} + 2.84 * E_{pm}^{ex} + 2.96 * E_{env}^{ex} + 0.65 \quad (51)$$

Zanimljiva činjenica koja se može primetiti je da je R funkcija za višestruku linearnu regresiju odbacila procene napora potrebnog za specifikaciju i funkcionalno testiranje dobijene na osnovu UCP veličine, zato što su ove vrednosti korelisane sa naporima potrebnim za definisanje opsega posla procenjenog na osnovu UCP veličine. Funkcija lm() pokušava da redukuje broj ulaznih dimenzija kako bi dobila formulu iste tačnosti. Parametri ovog modela su prikazani u tabeli 25.

Linearni model daje odlične procene sa MMRE greškom 9%, što je rezultat identičan Karolovom istraživanju [Carroll, 2005] koje predstavlja najbolji rezultat procene dobijen na osnovu UCP veličine. U ovom slučaju svi projekti imaju grešku procene manju od 25%, a PRED (25) je 1.

Tabela 25. Tačnost predikcionog modela dobijenog višestrukom regresijom.

Parametar	Višestruka regresija	UCP metoda	Ekspertska procena
MMRE	9%	23%	28%
PRED(25)	100%	62%	51%
R^2	96%	76%	69%
Min-Max(MRE)	0% - 21%	0% - 35%	0% - 66%

Sve procene projekta imaju greške prihvatljive prema PMI / Boemovim kriterijumima za rane procene. Pored toga, 82% projekata ima grešku u očekivanim granicama po kriterijumima za PMI budžetske procene.

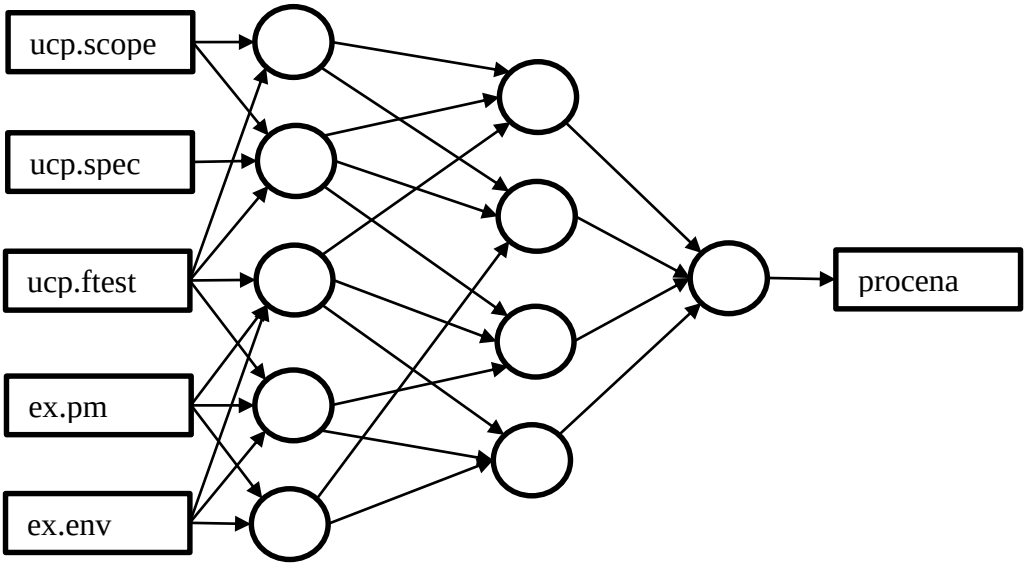
8.2.3.3. Zaključak

U ovom delu istraživanja je pokazano da model linearne regresije koji kao ulazne parametre koristi najoptimalnije tipove projektnih zadataka daje odlične rezultate procene napora.

8.2.4. Primena neuralnih mreža

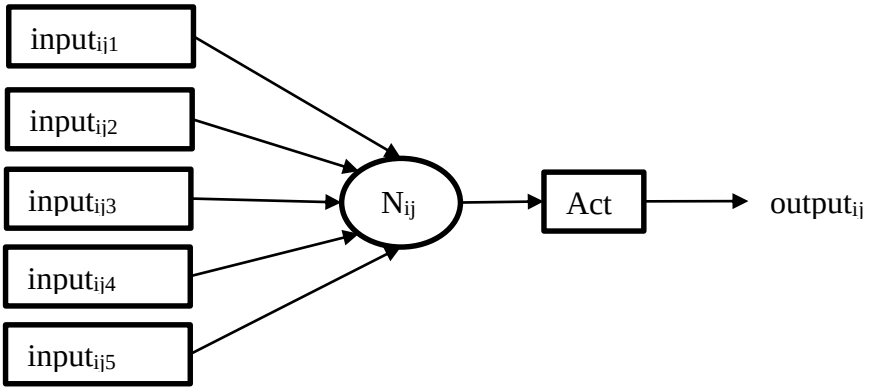
Linearna regresija je najjednostavnija funkcionalna zavisnost i to je jedan od glavnih razlog zašto se široko koristi radi procene napora. Međutim, potrebno je ispitati da li i druge funkcionalne zavisnosti kao što su polinomska, logaritamska ili eksponencijalna zavisnost mogu dovesti do još boljih rezultata. Provera svake moguće zavisnosti je neograničen proces a i diskutabilno je da li bi neka složenija funkcionalna zavisnost bila specifična baš za posmatrani skup projekata nad kojima se vrši analiza.

Umesto da se unapred definiše funkcionalna zavisnost, u ovom delu istraživanja se prepušta neuralnim mrežama [Jin *et al.*, 2012] da same na osnovu ulaza i izlaza odrede optimalnu funkcionalnu zavisnost. Neuralne mreže uče o podacima tokom faze obuke i dinamički prilagođavaju težinske koeficijente konekcija. Na slici 51 je prikazana neuralna mreža koja na osnovu pet ulaza određuje izlaz.



Slika 51. Procena napora pomoću neuralnih mreža.

Ulazi su optimalni tipovi projektnih zadataka dok je izlaz procenjeni napor. Neuroni u mreži imaju veze ka ostalim neuronima i svaka veza među neuronima ima svoju težinu. Konfiguracija jednog neurona se može predstaviti slikom 52.



Slika 52. Konfiguracija pojedinačnih neurona u mreži.

Svaki neuron N_{ij} koji se nalazi na i-toj poziciji u j-tom internom nivou ima k ulaznih veza (pet u primeru na slici 52) koje dovode podatke ili sa ulaza u neuralnu mrežu ili od neurona koji se nalaze u prethodnom sloju. Vrednosti na ulazu su obično u opsegu od 0 do 1. Svaka veza ima dodeljenu težinu w_{ijk}. Izlazna vrednost neurona koji se nalaze na poziciji i u nivou j se može izračunati formulom (52).

$$output_{ij} = \sum_{k} Act(w_{ijk} * input_{ijk} + bias) \tag{52}$$

Formula (52) je slična formuli za višestruku linearnu regresiju (50) pošto se k ulaznih vrednosti koje dolaze u neuron na poziciji i u sloju j ($output_{ijk}$) množe koeficijentima koji predstavljaju težine k-te veze ka neuronu (w_{ijk}) i sabiraju se kako bi se odredio izlaz neurona. Sumi se može dodati i konstantne vrednost (bias). Glavna razlika u odnosu na standardnu višestruku regresiju je u činjenici da se na ovu sumu primenjuje dodatna funkcija *Act* (aktivaciona funkcija u terminologiji neuralnih mreža) kojom se dobija izlazna vrednost iz neurona. Aktivaciona funkcija može da modifikuje sumu tako što propusti vrednost samo ako je suma veća od neke granice ili da primeni neku nelinearnu transformaciju pre slanja vrednosti na izlaz. U slučaju da je aktivaciona funkcija linearna funkcija sa koeficijentom 1, formula (52) se svodi na višestruku linearnu regresiju.

Postoje različite topologije neuralnih mreža. Najjednostavnije i najčešće korišćenje su mreže sa prebacivanjem podataka unapred (engl. *feedforward networks*) u kojima se podaci sa ulaza prosleđuju uvek u istom smeru kroz direktni aciklični graf bez povratnih veza. Uobičajeno, mreže imaju jedan ili više internih slojeva po kojima su raspoređeni neuroni (engl. *Multi-layered perception networks – MLP* [Rosenblatt 1958]) koji primaju informacije od neurona iz prethodnog nivoa ili ulaza u slučaju prvog nivoa, i prosleđuju izlaze neuronima iz sledećeg nivoa ili kao rezultat u slučaju poslednjeg nivoa.

Proces određivanja optimalnih težina veza među neuronima se naziva treniranje mreže. Uz predefinisanu topologiju mreže, broj internih slojeva i neurona po slojevima i odabrane aktivacione funkcije, na osnovu ulaza i očekivanih izlaza se menjaju težine veza dok se ne dobiju optimalne vrednosti. Treniranje mreže se vrši iterativno tako što se u svakoj iteraciji na ulaz mreže dovode ulazni podaci i porede dobijeni izlazi sa očekivanim vrednostima. Razlika između dobijenih i očekivanih izlaznih vrednosti predstavlja funkciju greške koja se prosleđuje nazad neuronima u mreži (engl. *backpropagation*) i ažuriraju se trenutne težine veza kako bi se smanjila greška.

Tokom procesa treniranja neuralne mreže, na ulaz se dovode vrednosti iz skupa za treniranje a na izlaz odgovarajući napor utrošen na projektu. Neuralna mreža tokom treniranja menja težine veza među neuronima kako bi se dobili optimalni koeficijenti za predviđanje izlaza u formuli (52).

Princip procene izlaza je sličan kao i kod linearne regresije (a i ostalih tipova regresija) pošto se ulazi množe nekim težinskim koeficijentima. Ti parcijalni rezultati se sabiraju, zatim im se dodaju odgovarajući težinski koeficijenti i na osnovu njih se računa rezultat. U zavisnosti od strukture mreže povratnih veza i koeficijenata koji su dobijeni, može se dobiti bilo koja nelinearna funkcionalna zavisnost kao rezultat.

8.2.4.1. Hipoteza

Trenirane neuralne mreže trebalo bi da identifikuju skrivene zavisnosti između procena napora projektnih zadataka i napora projekta i daju bolje procene napora.

8.2.4.2. Analiza

Procene napora na pojedinim tipovima projektnih zadataka (E_{tt} vrednosti korišćene u prethodnoj sekciji) i stvarni napor na projektu se koriste za obuku neuralne mreže.

Postoji veliki broj implementacija neuralnih mreža. U ovom delu istraživanja se višeslojna neuralna mreža (MLP) implementirana u R paketu *Stuttgart Neural Network* simulatora (RSNNS paket) [Bergmeir and Benítez, 2012]. Model neuralne mreže se dobija pozivom *mlp()* funkcije sledećom R naredbom:

```
nn.model <-mlp (task.types, effort)
```

Ulazni skup podataka se sastoji od istih ulaza koji se koriste u prethodnoj sekciji, tj. napora na projektnim zadacima koji se mogu najtačnije proceniti bilo UCP veličinama, bilo ekspertskim procenama. Ciljne vrednosti koje se dovode na izlaz mreže su odgovarajući napori na projektima koje treba proceniti.

Tokom inicijalizacije neuralne mreže potrebno je definisati njene karakteristike. Vrednosti neki parametara MLP mreže su fiksirane kao na primer:

1. Veze među neuronima u neuralnim mrežama su inicijalizovane slučajnim vrednostima u opsegu od -0.3 do 0.3. Ovo je preporučeno podešavanje MLP mreže u SNNS paketu u slučaju da ne postoji način da se unapred pretpostave težine veza.
2. Kao funkcija učenja se koristi standardno prosleđivanje parametara unazad (engl. *standard backpropagation*).

3. Prilikom ažuriranja neurona u MLP mreži se koristi topološki raspored neurona. Kao i u prethodnom slučaju ovo je podrazumevani algoritam u slučaju da ne postoji način da se pretpostavi koji redosled ažuriranja neurona bi trebalo koristiti.
4. Koristi se 100 iteracija prilikom treniranja mreža.
5. Sigmoidna funkcija prikazana u formuli (53) se koristi kao aktivaciona funkcija neurona. Ovo je preporučena funkcija u slučaju da nije moguće pretpostaviti koja je optimalna funkcija. Sigmoidna funkcija ne unosi promene u slučaju da se dobiju male ili srednje vrednosti grešaka, ali amortizuje veće vrednosti kako ne bi izazvale prevelike oscilacije prilikom treniranja.

$$sigmoid(x) = \frac{1}{1 - e^{x}} \tag{53}$$

Bitna odluka koju treba doneti je vezana za arhitekturu mreže, tj. broj nivoa i neurona po jednom nivou u mreži koji će biti korišćeni radi procene. S obzirom da nije moguće pronaći podrazumevane vrednosti broja slojeva i neurona, vrednosti ovih parametara su određene eksperimentalno. Radi procene optimalne konfiguracije u analizi je kreiran skup različitih konfiguracija MLP mreža koje su trenirane istim podacima. Potom su određene greške procene za svaku konfiguraciju kako bi se utvrdilo koja konfiguracija je optimalna.

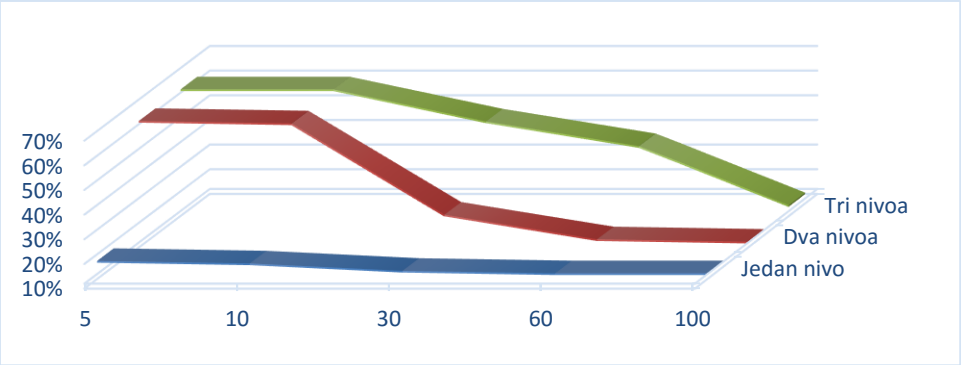
Na osnovu navedenih fiksnih parametara su generisane grupe MLP neuralnih mreža u kojima se menja broj nivoa u opsegu od 1 do 3 i broj neurona po nivou u opsegu od 5 do 100. Vrednosti grešaka procene i procenata projekata kojima je greška procene manja od 25% za neke karakteristične tačke su prikazane u tabeli 26.

Tabela 26. Tačnost predikcionog modela dobijenog treniranjem neuralne mreže.

	Broj nivoa					
Broj neurona po nivou	1		2		3	
	MMRE	PRED(25)	MMRE	PRED(25)	MMRE	PRED(25)
5	17%	77%	61%	25%	61%	25%
10	16%	80%	60%	28%	61%	25%
30	13%	92%	23%	64%	48%	25%
60	12%	92%	13%	92%	38%	60%
100	12%	92%	12%	92%	14%	80%

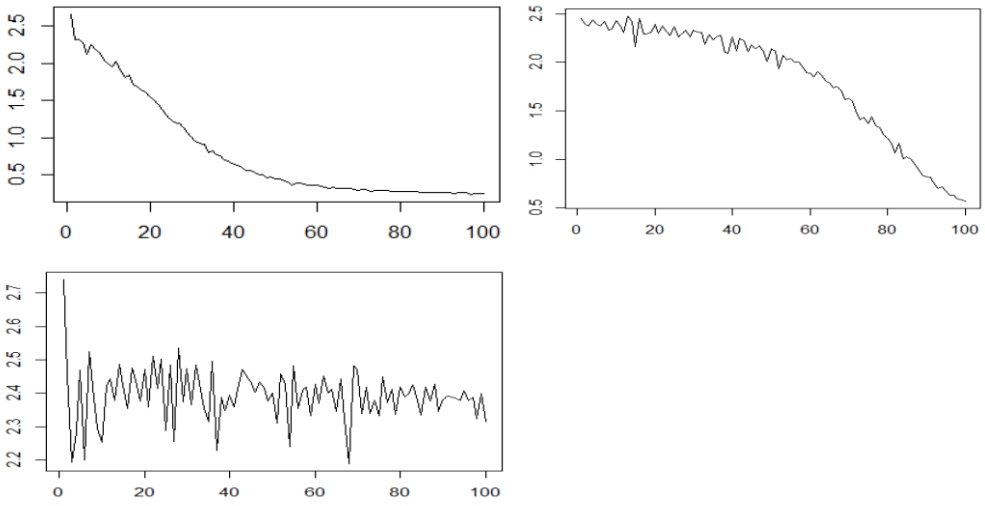
Kao što se i moglo očekivati, povećanjem broja neurona po nivou dobijaju se modeli sa boljim procenama napora. Međutim, smanjenje greške procene nije srazmerno povećanju broja neurona. Interesantna činjenica koja se može primetiti je da povećanje nivoa u ovom slučaju ne utiče na povećanje tačnosti modela. Najveću tačnost imaju MLP mreže sa po

jednim nivoom. Vrednosti greške procene u zavisnosti od broja čvorova su prikazane na slici 53.



Slika 53. Greške procene neuralnih mreža u zavisnosti od broja čvorova u internim (skrivenim) slojevima za mreže sa jednim, dva ili tri nivoa neurona.

Na osnovu slike se može zapaziti da je MLP mreža sa jednim skrivenim nivoom najoptimalnija u ovom slučaju. MLP mreže tokom svake iteracije u kojima se treniraju određuju grešku procene. Na slici 54 je prikazana promena grešaka tokom 100 iteracija treniranja mreža sa po 30 neurona po sloju i jednim, dva i tri interna sloja:



Slika 54. Promena greške po iteracijama tokom treniranja modela u zavisnosti od broj internih nivoa u neuralnoj mreži sa jednim, dva i tri nivoa.

Vrednost koja je prikazana na grafiku je suma kvadrata grešaka procene po svim projektima.

U MLP mrežama sa jednim slojem greške brzo konvergiraju od početnih vrednosti ka vrednostima manjim od 0.5. MLP mreža sa dva sloja sporije konvergira i krajnja greška ne pada ispod 0.5. U mreži sa tri sloja se mogu primetiti izrazite oscilacije grešaka koje čak i ne padaju ispod vrednosti 2.2. Bar na posmatranom skupu podataka može se zaključiti da su mreže sa jednim internim slojem optimalne za procenu napora.

Bez obzira na broj internih nivoa, može se primetiti da procene napora na osnovu neuralnih mreža konvergiraju ka vrednosti od 12% sa 92% projekata u kojima je greška procene manja od 25%.

Neuralna mreža sa jednim slojem i 60 neurona je odabrana radi kreiranja mreže kojom će se procenjivati napor na osnovu ulaznih veličina. Povećanje broja neurona u nivou ne doprinosi smanjenju greške, dok povećavanje broja nivoa unosi nestabilnost u procene.

Tokom treniranja mreže, ona će naći optimalne težine veza među neuronima kako bi se na osnovu ulaza iz skupa za treniranje dobile vrednosti najpribližnije ciljnim naporima projekta. Greške predikcionog modela dobijenog treniranjem neuralne mreže su prikazane u tabeli 27.

Tabela 27. Tačnost predikcionog modela dobijenog treniranjem neuralne mreže.

Parametar	Neuralne mreže	UCP metoda	Ekspertska procena
MMRE	12%	23%	28%
PRED(25)	92%	62%	51%
Min-Max(MRE)	0% - 39%	76%	69%

Iako se može primetiti da je procena napora na osnovu modela kreiranog pomoću neuralne mreže mnogo bolja od procene u originalnim metodama, rezultati ipak nisu bolji od višestruke linearne regresije, bar na posmatranom skupu podataka.

Razlog za lošiju procenu može da bude relativno mali broj projekata koji se koriste za treniranje neuralnih mreža koji nije dovoljan za kvalitetno treniranje mreže.

8.2.4.3. Zaključak

U ovoj sekciji je ispitano kako se mogu koristiti neuralne mreže radi kreiranja predikcionih modela koji daju bolje procenu napora na projektu od originalnih metoda. Dobijene procene su tri puta bolje od originalnih modela. Neuralne mreže omogućavaju

pronalaženje predikcionih modela bez ikakve a-priori pretpostavke o funkcionalnoj zavisnosti između ulaza i napora. Iako je greška procene niska i prihvatljiva, nije bolja od modela dobijenog višestrukom linearnom regresije.

8.2.5. Poređenje kombinovanih metoda procene

U tabeli 28 su prikazani parametri predikcionih modela opisanih u ovom poglavlju:

Tabela 28. Poređenje kombinovanih metoda procene.

	MMRE	PRED(25)	Min(Q0)	Q1	Q2	Q3	Max(Q4)	PMI-I	PMI-B	B-RS
UCP	23%	62%	0.00	0.09	0.19	0.35	0.66	90%	41%	82%
EXP	30%	51%	0.12	0.19	0.24	0.35	0.82	79%	10%	77%
SIN	19%	64%	0%	6%	14%	28%	52%	90%	51%	92%
IND	15%	85%	3%	6%	14%	22%	42%	97%	64%	97%
VLR	9%	100%	0%	5%	8%	12%	21%	100%	82%	100%
NN	12%	92%	0%	5%	8%	17%	39%	97%	72%	97%

U tabeli su uporedno prikazani parametri procene napora dobijeni na osnovu sledećih metoda:

1. UCP – procene napora na projektu dobijene linearnom regresijom i UCP veličinom sistema.
2. EXP – procene na projektu dobijene ekspertskim procenama.
3. SIN – procene napora na projektu dobijene kombinovanjem i sabiranjem procena projektnih zadataka dobijenih bilo UCP metodom ili ekspertskom procenom.
4. IND – procene napora na projektu dobijene na osnovu dela projektnih zadataka koji se najtačnije mogu proceniti bilo UCP metodom ili ekspertskom procenom, i procenom napora projekta na osnovu sume tih procena.
5. VLR – procene napora na projektu dobijene na osnovu dela projektnih zadataka koji se najtačnije mogu proceniti bilo UCP metodom ili ekspertskom procenom, i težinskim koeficijentima dobijenih višestrukom linearnom regresijom.
6. NN – procene napora na projektu dobijene na osnovu dela projektnih zadataka koji se najtačnije mogu proceniti bilo UCP metodom ili ekspertskom procenom i neuralnim mrežama.

Kriterijumi koji su prikazani za svaku metodu su greška procene (MMRE), procenat projekat sa greškama manjim od 25% (PRED(25)), kvartili, kao i broj projekata čije su greške procene u okviru očekivanih opsega po PMI inicijalnim procenama (PMI-I), PMI

budžetskim procenama (PMI-B) i po Boemovom procenama u trenutku definisanja zahteva (Boem-RS).

Posmatrajući rezultate može se primetiti da primena višestruke linearne regresije daje odlične rezultate procene. Pored poboljšanja procena u odnosu i na UCP i ekspertsku procenu, svi projekti zadovoljavaju PMI i Boemove kriterijume.

9. Zaključak

Predmet istraživanja predstavljenog u ovom poglavlju je unapređenje postojećih metoda procene napora na softverskim projektima kombinovanjem različitih subjektivnih metoda procene napora, metoda procene uticaja funkcionalnih i nefunkcionalnih zahteva, kao i naprednih predikcionih modela.

Dominantne metode procene napora u softverskim projektima su subjektivne procene kojima se na osnovu iskustva pojedinaca ili konsenzusom članova tima procenjuje napor na novom projektu. Neosporno je da subjektivne metode imaju dobrih strana posebno u slučajevima kada je potrebno koristiti intuiciju, heuristike ili poređenja po analogiji koja nije lako formulisati. Međutim, na ovaj način se ne iskorišćava u potpunosti dragoceno znanje o projektima koji su već implementirani u prošlosti.

Odsustvo dokazanih modela i algoritama za procenu napora kao i njihove primene u praksi trenutno pozicionira softversko inženjerstvo u vreme kada su ostale inženjerske grane bile zasnovane isključivo na iskustvu i intuiciji iskusnih inženjera koji su mogli da nađu rešenja za uvek nove i nepoznate probleme. Međutim, ostale inženjerske grane su u međuvremenu evoluirale i u potpunosti iskoristile podršku koje im pružaju naučne metode. Kao rezultat, može se primetiti ogroman korak napred u razvoju tehničkih rešenja u poslednjim decenijama. Primena naučnih metoda može pomoći softverskom inženjerstvu do dostigne sličan nivo zrelosti, posebno imajući u vidu da trenutne verovatnoće uspešnog završetka softverskih projekata nisu zadovoljavajuće [CHAOS, 2013].

U ovom radu je pokazano da i softversko inženjerstvo može izvući dosta korisnih informacija na osnovu naučnih metoda, bar kada se govori o domenu procene napora na projektima. Istraživanje sprovedeno u ovom radu je pokazalo kako se mogu identifikovati pravila kojima se dopunjuju postojeće metode procene napora.

Glavni rezultati kao što su metodologija kontinualne procene napora, kombinovanje metoda procene su predstavljeni u narednim sekcijama.

9.1. Metodologija procene napora tokom životnog ciklusa projekta

Prvi rezultat koji je dobijen u radu je potvrda da se metode procene zasnovane na veličinama projekata mogu uspešno koristiti radi procene napora. Greške procene predikcionih modela zasnovanih na ovim veličinama su u granicama očekivanja po kriterijuma definisanim od strane dr Berija Boema i PMI instituta. Ovi rezultati su bitni zato što dokazuju da se algoritamske metode mogu koristiti ravnopravno sa ekspertskim procenama. Jedino je potrebno uložiti dodatni napor radi prikupljanja istorijskih podataka potrebnih za analizu.

Bitan rezultat je predložena metodologija kontinualne procene napora tokom životnog ciklusa projekta kojom se u različitim fazama projekta mogu primeniti optimalne metrike radi procene napora na projektima. Ova metodologija je opisana u [Popovic and Bojic, 2012].

Primena ovakve metodologije ne isključuje mogućnost primene subjektivne procene napora. Ona je zapravo dodatak metodama subjektivne procene napora koji može da validira procene dobijene od strane projektnog tima ili da ukaže na potencijalne probleme usled neslaganja procena po različitim metodama tako da projektni timovi mogu da izvrše dodatnu analizu kako bi se pronašla bolja procena i korigovali rezultati što je pre moguće.

Bez dodatnih metoda, projektni timovi moraju da prihvate rizik da je jedina procena kojom se obavezuju njihovo subjektivno mišljenje bez obzira koliko su uvereni u njega.

9.2. Metode procene standardnih projektnih zadataka

Drugi bitan rezultat prikazan u istraživanju je činjenica da se tačnosti procene nekih standardnih projektnih zadataka (analiza zahteva, analiza i dizajn, testiranje) dobijene formalnim metoda ili procenama projektnih timove međusobno razlikuju po tačnosti procene [Popovic *et al.*, 2015]. Na ovaj način se mogu identifikovati projektni zadaci koji se mogu tačnije proceniti različitim metodama procene.

Ovaj rezultat omogućava projektnim timovima da naprave kratkoročne planove sa visokom pouzdanošću procena. Na ovaj način projektni timovi mogu da se sa velikom sigurnošću obavežu da će faze analize i dizajna završiti u predviđenim rokovima, dok će za ostale faze dati procene kada se sakupi više podataka tokom analize i dizajna.

Pored toga, ovaj rezultat otvara mogućnost kombinovanja različitih metoda procene napora na projektu tako što će se za svaki projektni zadatak primenjivati optimalna metoda procene za konkretni zadatak. Na ovaj način se kompenzuju nedostaci različitih metoda i primenjuju optimalna pravila procene.

9.3. Kombinovanje metoda za procenu napora

Konačno, najbitniji rezultat koji je predstavljen u ovom radu je potvrda da se metode subjektivne procene, metode za procenu veličine na osnovu funkcionalnih zahteva i metode za procenu uticaja nefunkcionalnih zahteva mogu međusobno kombinovati kako bi se dobila optimalna metoda za procenu napora.

Metode procene funkcionalnih i nefunkcionalnih zahteva su potpuno disjunktne, tako da projektni timovi imaju mogućnost da odaberu metode koje su najviše odgovaraju njihovima potrebama, kako po pitanju raspoloživosti informacija potrebnih za određene metode, tako i po osnovu korelacije procenjenih uticaja sa naporom na projektima.

Originalni rezultat predstavljen u ovom radu je aproksimativna metoda procene uticaja nefunkcionalnih zahteva kojom se mogu odbaciti parametri procene uticaja nefunkcionalnih zahteva iz postojećih metoda u slučaju da nemaju veliki uticaj na procene. Na ovaj način se ne predlaže još jedna metoda sa predefinisanim brojem parametara, već generalno pravilo za prilagođavanje postojećih metoda potrebama softverskih organizacija.

Originalni rezultat predstavljen u ovom radu je aproksimativna metoda procene nefunkcionalnih zahteva kojom je postojeća COCOMO II metoda uprošćena bez gubitka tačnosti procena.

Činjenica da se projektni zadaci mogu bolje ili lošije proceniti primenom različitih metoda je iskorišćen kao osnova za definisanje nove hibridne metode procene napora kojom se dobijaju rezultati bolji od rezultata dobijenih originalnim metodama procene napora. Hibridna metoda zasnovana na parcijalnim procenama projektnih zadataka otvara širok spektar mogućnosti za kombinovanje metoda procene – od direktnog kombinovanja ekspertskih i algoritamskih procena do korišćenja naprednih predikcionih modela kao što su standardna i višestruka linearna regresija, neuralne mreže i slično. Primena hibridnih

metoda unapređuje procene eksperata i standardnih algoritamskih metoda tri do četiri puta, bar u skupu podataka korišćenom u radu.

Kao rezultat, u radu je predstavljen skup metoda kojima se mogu unaprediti procene napora na projektima sa tačnošću boljom od tačnosti originalnih metoda.

Prilozi

Napor uložen na pojedinim projektnim zadacima

U tabeli 29 su predstavljene informacije o projektima koji su korišćeni u radu, napori uloženi na projektima kao i napori uloženi na pojedinim projektnim zadacima kao što su analiza zahteva (Req), analiza i dizajn (A&D), itd.

Tabela 29. Napor uložen na pojedinim projektnim zadacima.

Proj	Napor	Req	A&D	Const	Test	FTest	NFTtest	PM	SC	REQM	Spec	Env
P1	76	15	3.5	37	11	8	3	6	3	4	8	3.5
P2	48.5	12	2	22.5	7	5	2	4	3	2	7	1
P3	124.5	24	5	62.5	17	13	4	9	6	4	14	7
P4	51	11	2	24	8	6	2	4	3	2	6	2
P5	236.5	41.5	7	123.5	38	23	15	21	6	12	23.5	5.5
P6	58	16	2	23	9	5	4	5	3	3	10	3
P7	281	39	12	156	40	24	16	26	7	5	27	8
P8	81.5	12.5	6	45	9	6	3	7	3.5	2	7	2
P9	138	18	20	62	17	10	7	12	6	3	9	9
P10	59	14	1.5	29.5	9	7	2	4	3	2	9	1
P11	50	12.5	0	23	8	6	2	5	4	1.5	7	1.5
P12	24	3	1	12	5	4	1	2	1	1	1	1
P13	106	18	7	53	11	8	3	10	5	5	8	7
P14	100	25.5	5	46.5	13	10	3	8	4.5	7	14	2
P15	61	12	2	32	9	6	3	5	3	2	7	1
P19	110	26	4	53	16	12	4	9	6	5	15	2
P20	155	28	10	75	22	17	5	15	7	4	17	5
P21	210	37	12	105	27	21	6	21	9	5	23	8
P22	169.5	32	13	81	20.5	8	12.5	20	6	8	18	3
P24	129	28	5	62	19	15	4	10	7	7	14	5
P27	95	17	4	48	15	12	3	9	6	3	8	2
P29	225	34.5	8	125	32	20	12	22	7.5	7	20	3.5
P30	175	27	18	81	26	15	11	20	6	6	15	3
P31	161	28.5	8	86	18.5	12	6.5	16	6.5	5	17	4
P32	268.3	41	15	146	41	30	11	22	13	5	23	3.3
P34	210	29.5	12	115	30	20	10	18	5.5	2	22	5.5
P35	253.4	40	10	154	30	25	5	15	10	11	19	4.4
P37	90	20	5	42.5	13	8	5	8	4	8	8	1.5
P38	120	17	5	66	20	7	13	10	4	5	8	2
P39	61	13	2	32	8	6	2	5	4	2	7	1
P47	219.5	26	37	107.5	27	16	11	20	6	3	17	2
P49	110	23	2	58	14	10	4	8	5	6	12	5

Procene napora potrebnih radi kompletiranja projektnih zadataka

U tabeli su prikazana vremena koja je potrebno uložiti na pojedinim projektnim zadacima, procenjena od strane eksperata i projektnih timova.

Tabela 30. Procene napora potrebnih radi kompletiranja projektnih zadataka.

Proj	A&D	Const	FTest	NFTtest	PM	SC	REQM	Spec	Env	Procena
P1	5	25	7	2	6	3	2	7	4	61
P2	5	35	10	5	5	4	3	9	1	77
P3	5	45	10	4	10	5	3	13	5	100
P4	2.5	40	10	5	5	4	3	8	2.5	80
P5	5	85	21	12	20	5	7	11	5	171
P6	5	45	10	5	6	4	4	11	2	92
P7	10	120	20	16	24	12	9	29	6	246
P8	5	30	6	2	6	3	2	6	2	62
P9	10	45	7	3	10	6	4	10	6	101
P10	5	45	10	5	5	4	3	12	1	90
P11	5	45	15	10	6	6	4	9	2	102
P12	5	55	10	5	7	5	4	7	2	100
P13	5	40	7	2	10	4	3	6	6	83
P14	6	65	15	5	10	5	5	13	2	126
P15	5	32	7	7	6	5	5	10	1	78
P19	5	30	10	4	8	3	1	10	2.5	73.5
P20	6	100	25	10	13	10	6	20	2.5	192.5
P21	8	60	15	5	12	6	2	12	2.5	122.5
P22	10	65	8	4	18	5	5	15	3	133
P24	5	50	14	4	10	5	4	13	4	109
P27	5	30	10	2	10	3	1	10	2.5	73.5
P29	10	210	60	20	23	12	10	25	4	374
P30	10	68	10	10	18	5	5	17	2	145
P31	8	67	12	5	16	6	5	16	3	138
P32	10	100	28	14	25	11	7	26	4	225
P34	10	75	19	14	17	7	5	20	5	172
P35	15	180	30	10	25	10	9	25	5	309
P37	5	60	15	9	9	4	4	9	2	117
P38	4	55	9	5	10	4	4	11	2	104
P39	4	20	5	2	5	2	1	4	1	44
P47	10	170	43	20	24	9	7	22	2.5	307.5
P49	2	46	9	5	10	4	3	11	4	94

Funkcionalne veličine projekata određene UCP metodom

U tabeli 31 su prikazane karakteristike projekata koji su potrebne radi određivanja funkcionalne veličine softvera na osnovu UCP metode. Parametri koji se koriste su broj jednostavnih, srednjih i složenih slučajeva korišćenja odnosno korisnika sistema.

Tabela 31. Funkcionalne veličine projekata određene UCP metodom.

Proj	UC_L	UC_M	UC_H	A_L	A_M	A_H	#UC	UUCP	Napor
P1	4	2	0	2	0	1	6	45	76
P2	3	1	0	2	0	1	4	30	48.5
P3	5	3	1	2	0	1	9	75	124.5
P4	4	2	0	2	0	1	6	45	51
P5	9	5	2	2	0	1	16	130	236.5
P6	3	1	0	2	0	1	4	30	58
P7	11	7	2	2	0	1	20	160	281
P8	3	2	0	0	0	2	5	41	81.5
P9	4	3	1	0	0	2	8	71	138
P10	4	3	0	1	1	2	7	59	59
P11	3	1	1	0	1	1	5	45	50
P12	1	1	0	0	1	1	2	20	24
P13	3	2	0	0	1	1	5	40	106
P14	6	2	1	0	0	2	9	71	100
P15	4	3	0	0	1	1	7	55	61
P19	4	5	2	0	0	2	11	106	110
P20	8	6	3	0	0	2	17	151	155
P21	18	6	3	0	0	2	27	201	210
P22	4	3	0	0	2	2	7	60	169.5
P24	11	7	0	0	0	2	18	131	129
P27	5	5	1	0	1	1	11	95	95
P29	7	5	2	1	0	1	14	119	225
P30	5	5	2	0	0	2	12	111	175
P31	7	4	3	0	0	2	14	126	161
P32	8	8	6	1	0	3	22	220	268.3
P34	5	7	3	0	0	2	15	146	210
P35	5	6	4	1	0	3	15	155	253.4
P37	2	1	1	0	0	2	4	41	90
P38	2	1	0	0	0	2	3	26	120
P39	5	2	1	0	0	2	8	66	65
P47	8	5	1	0	0	2	14	111	219.5
P49	7	4	1	0	0	2	12	96	110

Karakteristike projekata dobijene NESMAI metodom

U tabeli 32 su prikazane karakteristike projekata (broj internih i eksternih fajlova) koje su potrebne radi određivanja funkcionalne veličine softvera na osnovu NESMA indikativne metode.

Tabela 32. Karakteristike projekata određene NESMA indikativnom metodom.

Proj	Nilf	Neif	Veličina	Napor
P1	4	4	200	76
P3	5	4	235	124.5
P5	9	4	375	236.5
P7	10	4	410	281
P8	2	3	115	45
P9	2	3	115	50
P10	3	4	165	59
P11	2	5	145	50
P12	1	5	110	24
P13	1	8	155	106
P14	6	3	255	100
P15	3	3	150	61
P16	2	6	160	75
P17	3	0	105	40
P19	7	0	245	110
P20	9	0	315	155
P21	9	0	315	210
P23	5	0	175	60
P24	7	0	245	129
P27	6	0	210	95
P30	*8*	0	*280*	175
P35	*17*	*0*	*595*	253.4
P38	2	5	145	120
P40	4	0	140	57
P48	4	0	140	84
P49	7	0	245	110

Karakteristike projekata dobijene Mark II metodom

U tabeli 33 su prikazani brojevi ulaza, izlaza i objekata u sistemu koji se koriste radi procene napora Mark II metodom. Veličine projekata su određene na osnovu ovih veličina.

Tabela 33. Karakteristike projekata određene Mark II metodom.

Proj	Ni	Ne	No	Veličina	Napor
P1	17	15	8	36.84	76
P3	19	23	9	51.54	124.5
P5	36	39	13	89	236.5
P7	41	45	14	102.12	281
P8	5	7	5	15.82	45
P9	6	10	5	21.38	50
P10	7	10	7	22.48	59
P11	9	9	6	21.72	50
P12	4	6	3	10.16	24
P13	4	13	8	25.98	106
P14	20	18	9	43.82	100
P15	9	11	6	25.04	61
P16	5	14	8	28.22	75
P17	6	9	3	19.2	40
P19	14	19	7	41.48	110
P20	27	17	9	46.22	155
P21	27	31	9	69.46	210
P23	10	14	5	30.34	60
P24	20	24	7	53.26	129
P30	19	24	8	52.94	175
P40	11	12	4	27.34	57
P48	3	16	4	29.34	84
P49	18	20	7	45.46	107

Karakteristike projekata dobijene NESMAE metodom

U tabeli 34 su prikazani brojevi ulaza, upita, izlaza, internih i eksternih fajlova u projektima koji su korišćeni u analizi radi procene napora NESMA procenjenom metodom.

Tabela 34. Karakteristike projekata određene NESMA procenjenom metodom.

Proj	N-EI	N-EQ	N-EO	N-ILF	N-EIF	Veličina	Napor
P1	13	18	7	4	4	207	76
P3	18	15	9	5	4	232	124.5
P5	35	31	9	9	4	392	236.5
P7	40	37	9	10	4	443	281
P8	5	6	2	2	3	83	45
P9	5	6	5	2	3	98	50
P10	6	8	3	3	4	112	59
P11	10	6	2	2	5	113	50
P12	4	7	1	1	5	81	24
P13	4	12	1	**1**	8	116	106
P14	19	16	3	6	3	212	100
P15	8	10	2	3	3	118	61
P16	4	14	1	2	6	121	75
P17	6	8	1	3	0	82	40
P19	10	17	1	7	0	162	110
P20	25	14	6	9	0	249	155
P21	25	14	20	9	0	319	210
P23	9	9	6	**5**	0	137	60
P24	18	23	2	7	0	223	138
P27	14	13	1	6	0	155	100
P30	19	15	9	***8***	0	237	175
P40	10	10	3	4	0	123	57
P48	3	11	5	4	0	109	84
P49	17	19	3	7	0	208	110

Karakteristike projekata dobijene FPA metodom

U tabeli 35 su prikazane informacije o broju funkcionalnosti i fajlova, kao i veličine sistema izražene u broju neprilagođenih funkcionalnih tačaka koje su određena na osnovu njih.

Tabela 35. Karakteristike projekata određene metodom funkcionalnih tačaka.

Proj	UFP-T	UFP-F	Veličina	Napor
P1	103	60	163	76
P3	150	74	224	124.5
P5	250	107	357	236.5
P7	290	114	404	281
P8	50	29	79	45
P9	56	31	87	50
P10	58	51	109	59
P11	63	44	107	50
P12	37	36	73	24
P13	52	55	107	106
P14	120	63	183	100
P15	62	39	101	61
P16	64	46	110	75
P17	48	34	82	40
P19	100	76	176	110
P20	170	100	270	155
P21	194	100	294	210
P24	140	93	233	129
P27	85	75	160	100
P30	185	91	276	175
P40	72	54	126	57
P48	62	44	106	64
P49	130	80	210	110

Težinski faktori kojima se određuje uticaj NFZ COCOMO II metodom

U tabeli 36 su prikazane vrednosti parametara kojima se ocenjuje uticaj nefunkcionalnih zahteva u COCOMO II metodi.

Tabela 36. Težinski faktori koji se dodeljuju COCOMO II parametrima.

Parametar	Simbol	VL	L	N	H	VH	XH
PREC	SF_1	0.05	0.04	0.03	0.02	0.01	0
FLEX	SF_2	0.05	0.04	0.03	0.02	0.01	0
RESL	SF_3	0.05	0.04	0.03	0.02	0.01	0
TEAM	SF_4	0.05	0.04	0.03	0.02	0.01	0
PMAT	SF_5	0.05	0.04	0.03	0.02	0.01	0
RELY	EM_1	0.75	0.88	1	1.15	1.4	
DATA	EM_2		0.94	1	1.08	1.16	
CPLX	EM_3	0.75	0.88	1	1.15	1.3	1.65
RUSE	EM_4		0.89	1	1.16	1.34	1.56
DOCU	EM_5	0.85	0.93	1	1.08	1.17	
TIME	EM_6			1	1.11	1.3	1.66
STOR	EM_7			1	1.06	1.21	1.56
PVOL	EM_8		0.87	1	1.15	1.3	
ACAP	EM_9	1.5	1.22	1	0.83	0.67	
PCAP	EM_{10}	1.37	1.16	1	0.87	0.74	
PCON	EM_{11}	1.26	1.11	1	0.91	0.83	
AEXP	EM_{12}	1.23	1.1	1	0.88	0.8	
PEXP	EM_{13}	1.26	1.12	1	0.88	0.8	
LTEX	EM_{14}	1.24	1.11	1	0.9	0.82	
TOOL	EM_{15}	1.2	1.1	1	0.88	0.75	
SITE	EM_{16}	1.24	1.1	1	0.92	0.85	0.79
SCED	EM_{17}	1.23	1.08	1	1.04	1.1	

Uticaji NFZ određeni FPA, UCP, COCOMO II metodom

U tabeli 36 su prikazani parametri kojima se kvantifikuje uticaj nefunkcionalnih zahteva određen pomoću FPA, UCP, COCOMO II, i aproksimativne metode predstavljene u sekciji 6.2.

Tabela 37. Karakteristike projekata određene COCOMO II metodom.

	FPA	UCP	COCOMO II		Aproksimativna
Proj	VAF	AF	AF	SF	K
P1	1	0.81375	0.552169	1.156	0.815728
P3	1.01	0.8372	0.689866	1.156	1.01915
P5	1.04	0.84165	0.926091	1.156	1.192406
P7	1.05	0.8507	0.799806	1.156	1.084005
P8	1.03	0.7832	0.519688	1.156	0.704352
P9	1.04	0.7921	0.441735	1.156	0.598699
P10	1.05	0.8372	0.601563	1.156	0.81532
P11	1.05	0.84165	0.401979	1.156	0.544816
P12	1.05	0.83125	0.414443	1.156	0.533623
P13	1.05	0.827475	0.528391	1.156	0.68034
P14	1.03	0.80545	0.737825	1.156	1
P15	1.04	0.791875	0.643705	1.156	0.872436
P16	1.03	0.827475	0.630168	1.156	0.854088
P17	1.04	0.7611	0.460525	1.156	0.68034
P19	1.05	0.7611	0.643705	1.156	0.872436
P20	1.01	0.791875	0.695122	1.156	0.942123
P21	1.02	0.791875	0.79899	1.156	1.0829
P23	1.03	0.7611	0.501972	1.156	0.68034
P24	1.01	0.808775	0.751954	1.156	1.01915
P25	1.04	0.77875	0.683595	1.156	0.9265
P27	1.01	0.7743	0.643705	1.156	0.872436
P29	1	0.902275	0.804229	1.156	1.09
P30	1.01	0.8464	1.035043	1.156	1.40283
P32	1.02	0.8234	0.707722	1.156	1.09
P33	0.99	0.7964	0.879786	1.156	1.192406
P34	1	0.7832	0.751954	1.156	1.01915
P35	1	0.8096	0.751954	1.156	1.01915
P37	1.01	0.9212	0.910498	1.156	1.23403
P38	1.06	0.851	1.149317	1.156	1.55771
P40	0.99	0.805	0.615718	1.156	0.834504
P48	1.02	0.827475	0.924863	1.156	1.09
P49	1.03	0.7482	0.723806	1.156	1.09

Literatura

[Abran *et al.*, 1999] A. Abran, J. Desharnais, S. Oligny, D. St-Pierre, C. Symons, „COSMIC-FFP Measurement Manual, version 2.0", Ed. S. Oligny, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Montreal, Canada, October 1999.

[Abran *et al.*, 2004] A. Abran, R. Meli, C. Simons, „COSMIC Full Function Point Method: 2004 – State of Art", SMEF04, Rome, Italy, January 2004.

[Aho *et al.*, 1986] A. Aho, R. Sethi, J. Ullman, „Compilers: Principles, Techniques, and Tools", Addison-Wesley 1986, ISBN 0-201-10088-6.

[Albrecht and Gaffney, 1983] A. Albrecht, J. Gaffney, „Software Function, Source lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, SE-9 (6): pp. 639-648, 1983.

[Albrecht, 1994] A. Albrecht, „Software Development Cost Estimation Using Functional Point", IEEE Transactions on Software Engineering, April 1994.

[Allen, 1970] F. E. Allen, „Control flow analysis", Proc. Symp. on Compiler Optimization, pp. 1-19, 1970.

[Allen, 1974] F. E. Allen, „Interprocedural data flow analysis", in Proceedings of the IFIP Congress, pp. 398–402, 1974.

[Alves *et al.*, 2013] Alves, L., Sousa, A., Ribeiro, P., et al.: 'An Empirical Study on the Estimation of Software Development Effort with Use Case Points', in 43rd annual Frontiers in Education Conference (IEEE), pp. 101-107, Oklahoma City, Oklahoma, USA, October 2013.

[Azzeh, 2013] Azzeh, M.: 'Fuzzy Model Tree for Early Effort Estimation Machine Learning and Applications', 12th International Conference on Machine Learning and Applications, pp. 117-121, Miami, Florida, USA, 2013.

[Beck, 1999] K. Beck, „Extreme Programming Explained: Embrace Change”, Addison-Wesley Professional, 1999.

[Bergmeir and Benítez, 2012] C. Bergmeir, J.M. Benítez, „Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS”. Journal of Statistical Software, Vol. 46, No. 7, pp. 1-26. 2012, http://www.jstatsoft.org/ v46/i07/

[Boehm *et al.*, 2000] B. Boehm, C. Abts, A. Winsor Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, B. Steece, „Software Cost Estimation with COCOMO II”, Prentice-Hall, New Jersey, USA, 2000, ISBN-10: 0137025769.

[Booch *et al.*, 1998] G. Booch, J. Rumbaugh, I. Jacobson, „The Unified Modeling Language User Guide”, Addison-Wesley, 1998.

[Briand *et al.*, 1997] Lionel C. Briand, Khaled El Emam, Frank Bomarius, „COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment”, International Software Engineering Research Network Technical Report, ISERN-97-24, 1997.

[Carroll, 2005] Carroll, E. R.: 'Estimating software based on use case points', Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 257-265, San Diego, CA, USA, October 16-20, 2005.

[CHAOS, 2013] CHAOS Report, The Standish Group, 2013.

[Clemmons, 2006] R. Clemmons, „Project Estimation With Use Case Points”, The Journal of Defense Software Engineering, 2006.

[CMMI, 2012] CMMI Product team: 'CMMI for Development, Version 1.3', Carnegie Mellon Software Engineering Institute, CMU/SEI-2010-TR-033, November 2012.

[Cockburn, 2001] A. Cockburn, „Writing Effective Use Cases”, Addison-Wesley, 2001.

[Constantine and Lockwood, 1999] L. Constantine, L. Lockwood, „Software for Use”, ACM Press, Addison Wesley, 1999.

[Conte *et al.*, 1986] S.D. Conte, H.I. Dunsmore, V.Y. Siien, : ‘Software engineering metrics and models’, Benjamin/Cummings, Menlo Park, CA,1986.

[Costangiola and Tortora, 2005] G. Costagliola, G. Tortora, „Class Point: An Approach for the Size Estimation of Object-Oriented Systems”, IEEE Transactions on Software Engineering, Vol. 31, No.1, January 2005.

[Cvetkovic *et al.*, 2004] D. Cvetković, I. Lacković, M. Merkle, Z. Radosavljević, S. Simić, P. Vasić, Matematika I - Algebra, VIII izdanje, Akademska misao, Beograd, 2004.

[DeMarco, 1978] T. DeMarco, „Structured Analysis and System Specification”, Prentice-Hall, 1978.

[Dwyer, 1997] M. Dwyer, Modular flow analysis for concurrent software, Proceedings of the 12th international conference on Automated software engineering, pp. 264-273, Incline Village, NV, USA, November 01-05, 1997.

[Foss *et al.*, 2003] T. Foss, E. Stensrud, B. A. Kitchenham, : 'A Simulation Study of the Model Evaluation Criterion MMRE', IEEE Transactions on Software Engineering, Vol. 29, pp. 985-995, 2003.

[Fox and Weisberg, 2011] J. Fox, S. Weisberg, : 'An {R} Companion to Applied Regression', Second Edition. Thousand Oaks CA: Sage Publications, 2011, URL: http://socserv.socsci.mcmaster.ca/jfox/Books/Companion

[Frohnhoff and Engels, 2008] S. Frohnhoff, G. Engels: 'Revised use case point method - effort estimation in development projects for business applications', in 11th International Conference on Quality Engineering in Software Technology (CONQUEST 2008), pp. 15-32, Potsdam, Germany,24–26 September, 2008.

[Garmis and Herron, 2001] D. Garmis, D. Herron, „Functional Point Analysis“, Addison-Wesley, Boston USA, 2001.

[Gelperin, 2004] D. Gelperin, „Precise Use Cases” LiveSpecs Software, 2004.

[González-Carrasco *et al.*, 2012] I. González-Carrasco, R. Colomo-Palacios, J. L. López-Cuadrado, F. Peñalvo: 'SEffEst: Effort estimation in software projects using fuzzy logic

and neural networks', International Journal of Computational Intelligence Systems, 5, (4), pp. 679-699, 2012.

[GSHelp] GoogleSpreadsheets help, http://docs.google.com/support/

[Halstead, 1997] M. Halstead, „Elements of Software Science", Elsevier North-Holland, New York, USA, 1997.

[Han and Kamber, 2006] J. Han, M. Kamber, „Data Mining: Concepts and Techniques", Second Edition, The Morgan Kaufmann Series in Data Management Systems, 2006.

[Harrell, 2001] F. Harrell, Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis. New York: Springer; 2001

[Harrold and Rothermel, 1994] M. J. Harrold and G. Rothermel, „Performing data flow testing on classes," SIGSOFT '94 Proceedings of the 2nd ACM SIGSOFT symposium on Foundations of software engineering, pp. 154-163, New Orleans, Louisiana, United States, 1994.

[ISOFSM, 2007] ISO/IEC 14143-1:2007 Information technology – Software measurement – Functional size measurement.

[IFPUG, 1994] „Function Point Counting Practices Manual, Release 4.0", International Function Point User Group, Westerville, Ohio, Jan. 1994.

[Jin *et al.*, 2012] C. Jin, S. W. Jin, J. M. Ye: 'Artificial neural network-based metric selection for software fault-prone prediction model', IET software, 6, (6), pp. 479-487, 2012.

[Jorgensen *et al.*, 2009] M. Jorgensen, B. Boehm, S. Rifkin, 'Software Development Effort Estimation: Formal Models or Expert Judgment?', Software, IEEE, Volume:26 , Issue: 2, pp. 14-19, 2009.

[Jorgensen, 2010] M. Jorgensen: Selection of strategies in judgment-based effort estimation, Journal of Systems and Software, Vol 83, No 6, pp.1039-1050, 2010.

[Jorgensen, 2014] M. Jorgensen: What We Do and Don't Know about Software Development Effort Estimation, IEEE Software, Volume:31 , Issue: 2, pp 37 – 40, ISSN: 0740-7459.

[Kamal *et al.*, 2011] M. Kamal, M. Ahmed, M. El-Attar: 'Use Case-Based Effort Estimation Approaches: A Comparison Criteria', Second International Conference Software Engineering and Computer Systems ICSECS 2011, pp. 735-754, Kuantan, Pahang, Malaysia, June 27-29, 2011.

[Kassab *et al.*, 2014] M. Kassab, C. Neill, P. Laplante: 'State of practice in requirements engineering: contemporary data', Innovations in Systems and Software Engineering, 10, (4), pp. 235-241, April 2014.

[Karner, 1993] G. Karner, „Use Case Points - Resource Estimation for Objectory Projects", Objective Systems SF AB, 1993.

[Kemerer, 1987] C. Kemerer, „An empirical validation of software cost estimation models", Communications of the ACM, Vol. 30, No.5, p.416-429, May 1987.

[Kemerer and Porter, 1992] C.F. Kemerer, B.S. Porter. „Improving the Reliability of Function Point Measurement: An Empirical Study. IEEE Transactions on Software Engineering, 18(11), pp. 1011–1024, 1992.

[Kitchenham *et al.*, 2001] B.A. Kitchenham, S.G. MacDonell, L.M. Pickard: 'What Accuracy Statistics Really Measure', IEEE Proc. Software, 148, (3), pp. 81-85, 2001.

[Kocaguneli and Mensies, 2013] E. Kocaguneli, T. Menzies: 'Software effort models should be assessed via leave-one-out validation', Journal of Systems and Software, 86, (7), pp. 1879-1890, July 2013.

[Kruchten, 2000] R. Kruchten „The Rational Unified Process – An Introduction", Addison-Wesley, Boston, USA, 2000.

[Kutner *et al.*, 2004] M. Kutner, C. Nachtsheim, J. Neter: 'Applied Linear Regression Models' 4th Edition, McGraw-Hill Education (January 8, 2004), ISBN-13: 978-0073014661.

[Kusumoto *et al.*, 2004] S. Kusumoto, F. Matukawa, I. Katsuro, „Estimating Effort by Use case Points: Method, Tool and Case Study“, 10th International Symposium on Software Metrics, 2004.

[Lavazza and Robiolo, 2010] L. Lavazza, G. Robiolo: 'The role of the measure of functional complexity in effort estimation', in PROMISE '10 Proceedings of the 6th International Conference on Predictive Models in Software Engineering, Article No. 6, Timişoara, Romania,September 12-13, 2010.

[Lavazza and Robiolo, 2012] L. Lavazza, G. Robiolo: 'Using Functional Complexity Measures in Software Development Effort Estimation', International Journal on Advances in Software, Vol. 5, (3&4), pp. 263-277, 2012.

[Larman, 2001] C. Larman, „Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development”, Prentice-Hall, New Jersey, USA, 2001.

[Lind and Vairavan, 1989] K. Lind, K. Vairavan, „An experimantal investigation of software metrics and their relationship to software development effort", IEEE Trans. Software Engineering, Vol. SE-15, pp. 649-653, May 1989.

[Lockwood and Lockwood, 1999] C. Lockwood, L. Lockwood, „Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design", Reading, MA: Addison-Wesley, 1999.

[McCabe, 1976] T. McCabe, „A Complexity Measure". IEEE Transactions on Software Engineering, Volume:SE-2 , Issue: 4, pp. 308–320. December 1976.

[Mendes, 2008] E. Mendes: The use of Bayesian networks for web effort estimation: further investigation. In Proceedings of Eighth International Conference on Web engineering ICWE'08, IEEE, pp. 203-216, July 2008.

[Merkle i Vasic, 1997] M. J. Merkle, P.M. Vasic: Verovatnoca i statistika - sa primenama i primerima, Elektrotehnicki fakultet, Beograd, 1995, II izdanje 1997. xii+307.

[MSF, 2003] „Microsoft Solutions Framework version 3.0 Overview“, White paper, 2003, Microsoft.

[MSF, 2008] „Preparation Guide for Exam 70-300: Analyzing Requirements and Defining Microsoft .NET Solution Architectures, Microsoft, September 2008.

[MSOffice] Microsoft Office help, http://office.microsoft.com/

[MDX, 2012] 'Multidimensional Expressions (MDX) reference', SQL Server Books online, Microsoft, June 2012.

[Mulcahy, 2013] R. Mulcahy, "PMP Exam Prep, Eighth Edition: Rita's Course in a Book for Passing the PMP Exam," RMC Publications; Eighth edition, June 12, 2013.

[Nagelkerke, 1991] N. Nagelkerke: A note on a general definition of the coefficient of determination. Biometrika, (78), pp. 691–692, 1991.

[Nassif *et al.*, 2010] A.B. Nassif, L.F. Capretz, D. Ho: 'Enhancing Use Case Points Estimation Method using Soft Computing Techniques', Journal of Global Research in Computer Science, Vol. 1, No. 4, pp. 12- 21, November 2010.

[Nassif *et al.*, 2011] A.B. Nassif, L.F. Capretz, D. Ho: 'Estimating software effort based on use case point model using sugeno fuzzy inference system', in 23rd IEEE International Conference on Tools with Artificial Intelligence, pp. 393-398, Florida, USA, 2011.

[Nassif, 2012] A.B. Nassif: 'Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models'. PhD thesis, Western University, 2012.

[Nassif *et al.*, 2012] A.B. Nassif, L.F. Capretz, D. Ho: 'A treeboost model for software effort estimation based on use case points', in 11th International Conference on Machine Learning and Applications (ICMLA), pp. 314-319, Boca Raton, Florida, USA, 2012.

[Nassif *et al.*, 2013] A.B. Nassif, D. Ho, L.F. Capretz: 'Towards an early software estimation using log-linear regression and a multilayer perceptron model', Journal of Systems and Software, Vol. 86, No. 1, pp. 144–160, January 2013.

[Niessink, 1997] F. Niessink, „Predicting Maintenance Effort with Function Points", in the Proceedings of ICSM'97, September 28 – October 2, 1997.

[NESMA, 1997] Netherlands Software Metrics Association (NESMA), „Definition and Counting Guidelines for the Application of Function Point Analysis”, Version 2.0, Amsterdam NESMA 1997.

[NetoAlvarez, 2003] S. NetoAlvarez, „PROJECT MANAGEMENT FAILURE: MAIN CAUSES”, Bowie State University, Maryland in Europe, March 2003.

[Nuzzo, 2014] R. Nuzzo, "Scientific method: Statistical errors". Nature 506 (7487): 150, 2014, doi:10.1038/506150a.

[Ochodek and Nawrocky, 2010] M. Ochodek, J. Nawrocki: 'Enhancing Use-Case-Based Effort Estimation with Transaction Types', Foundations of Computing and Decision Sciences, Vol. 35, No. 2, pp. 91–106, 2010.

[Ochodek *et al.*, 2011] M. Ochodek, J. Nawrocki, K. Kwarciak: 'Simplifying effort estimation based on Use Case Points', in Journal Information and Software Technology archive, Vol. 53, No. 3, pp. 200-213, March 2011.

[Park, 1992] R. Park. „Software size measurement: a framework for counting source statements” CMU/SEI-92-TR-020.

[Parvez, 2013] A. Parvez: 'Efficiency Factor and Risk Factor Based User Case Point Test Effort Estimation Model Compatible with Agile Software Development', In Proceedings of International Conference on Information Technology and Electrical Engineering, pp. 113-118, Yogyakarta, Indonesia, 2013.

[PMBok, 2004] „A Guide to the Project Management Body of Knowledge (PMBOK® Guide)”, Project Management Institute, 2004, ISBN 978-1-933890-51-7.

[Popovic, 2010] J. Popović: „Merenje i analiza softverskih sistema“, Magistarska teza, Elektrotehnički fakultet, Univerzitet u Beogradu, 2010.

[Popovic and Bojic, 2012] J. Popović, D. Bojić: „A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle“, Computer Science and Information Systems, Vol. 9, No. 1, pp. 455-484, 2012, ISSN: 1820-0214.

[Popovic, Puric, *et al.*, 2015] J. Popović, S. Purić, M. Batić, D. Bojić: „Enhancing Use Case Point estimation method using Fuzzy algorithms“, Zbornik radova sa 23. Telekomunikacionog foruma TELFOR 2015, 2015.

[Popovic *et al.*, 2015] J. Popović, D. Bojić, N. Korolija: „Analysis of task effort estimation accuracy based on use case point size“, IET Software, 11/2015; DOI:10.1049/iet-sen.2014.0254.

[Putnam, 1997] L. Putnam, „A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-4, NO. 4, JULY 1978.

[Putnam *et al.*, 2005] L. Putnam, D. Putnam, and D. Beckert, „A Method for Improving Developers' Software Size Estimates," Crosstalk, Vol. 18, No. 4, pp. 16–19, 2005.

[R, 2014] R Core Team: 'R: A language and environment for statistical computing', R Foundation for Statistical Computing, Vienna, Austria, 2014.

[Reifer, 2000] D. Reifer, „Web Development: Estimating Quick-To-Market Software," IEEE Software, Vol. 17, pp. 57-64, 2000.

[Rosenblatt 1958] F. Rosenblatt F, „The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." Psychological Review, Vol. 65, No. 6, pp. 386–408.

[Ruhe *et al.*, 2003] M. Ruhe, R. Jeffery, I. Wieczorek, „Using Web Objects for Estimating Software Development Effort for Web Applications", Proceedings of the 9th International Symposium on Software Metrics, p. 30, September 03-05, 2003.

[Rumbaugh, 1994] J. Rumbaugh, „Getting Started : Using Use Cases to Capture Requirements" Journal of OO Programming, SIGS Publications, Vol. 7, No. 5, Sept 1994.

[Sakamoto *et al.*, 1986] Y. Sakamoto, M. Ishiguro, G. Kitagawa: 'Akaike Information Criterion Statistics', D. Reidel Publishing Company, 1986.

[Schofield *et al.*, 2013] J. Schofield, A. Armemtrout, R. Trujillo: 'Function Points, Use Case Points, Story Points — Observations From a Case Study', CrossTalk – the Journal of Defense Software Engineering, Vol. 26, No. 4, pp. 23 – 27, May/June 2013.

[Schwaber and Beedly, 2001] K. Schwaber, M. Beedly, „Agile Software Development with Scrum", Prentice Hall, 2001.

[Shepperd and MacDonell, 2012] M.J. Shepperd, S.G. MacDonell: 'Evaluating prediction systems in software project estimation', Information & Software Technology, Vol. 54, No. 8, pp. 820–827, August 2012.

[Smith, 2003] J. Smith, „The Estimation of Effort Based on Use Cases", White paper, IBM Rational Software, 2003.

[Stain *et al.*, 2005] C. Stein, G. Cox, L. Etzkorn, „Exploring the relationship between cohesion and complexity". Journal of Computer Science, Volume 1, Issue 2, Pages 137-144, 2005.

[St-Pierre *et al.*, 1997] D. St-Pierre, M. Maya, A. Abran, J.M. Desharnais , P. Bourque, „Full Function Points: Function Points Extension for Real-Time Software - Counting Practices Manual", Technical Report no. 1997-04, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Montreal, Canada, September 1997.

[Subriadi and Ningrum, 2014] A.P. Subriadi, P.A. Ningrum: 'Critical review of the effort rate value in use case point method for estimating software development effort', Journal of Theoretical and Applied Information Technology, Vol. 59, No. 3, pp. 735–744, 2014.

[Symons, 1991] C. Symons, „Software Sizing and Estimating, Mk II Function Point Analysis", John Wiley & Sons, Chichester, England, 1991.

[Symons and Rule, 1999] C.R. Symons, P. G. Rule: „One size fits all – COSMIC aims, design principles and progress", Proceedings of ESCOM '99, pp. 197-207, April 1999.

[Schervish, 1996] M.J. Schervish: "P Values: What They Are and What They Are Not". The American Statistician, Vol. 50, No. 3. doi:10.2307/2684655. JSTOR 2684655.

[Usman *et al.*, 2014] M. Usman, E. Mendes, F. Weidt F., Brito R.: Effort estimation in agile software development: a systematic literature review, In Proceedings of the 10th International Conference on Predictive Models in Software Engineering, PROMISE ’14, pp. 82-91, ACM New York, 2014.

[UKSMA, 1998] „MK II Function Point Analysis: Counting Practices Manual, Version 1.3.1.” United Kingdom Software Metrics Association (UKSMA), Sept 1998.

[Urbanek *et al.*, 2014] T. Urbanek, Z. Prokopova, R. Silhavy, S. Sehnalek: 'Using Analytical Programming and UCP Method for Effort Estimation', Modern Trends and Techniques in Computer Science, Advances in Intelligent Systems and Computing, Vol. 285, pp. 571-581, May 2014,

[Venables and Ripley, 2002] W.N. Venables, B.D. Ripley: 'Modern Applied Statistics with, S', Fourth Edition. Springer, New York, 2002, ISBN 0-387-95457-0.

[Walston and Felix, 1977] C. E. Walston, C. P. Felix, „A method of programming measurement and estimation, IBM Systems Journal”, Vol. 16, No. 1, pp. 54-73, 1977.

[Watson and McCabe, 1996] A. Watson, T. McCabe: „Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric”, NIST Special Publication 500-235, 1996.

[Zuse, 2005] H. Zuse, „Resolving the Mysteries of Halstead Measures“, Technische Universität Berlin, November 2005.

Podaci o autoru:

Jovan Popović je rođen 8.12.1980. u Pančevu, republika Srbija, gde je završio osnovnu i srednju školu sa odličnim uspehom. Za vreme osnovnog i srednjeg školovanja učestvovao i osvajao nagrade na mnogobrojnim takmičenjima iz matematike, fizike, elektrotehnike i elektronike, među kojima četiri puta jedno od prva tri mesta na nacionalnim takmičenjima. Bio je u užem krugu kandidata za olimpijadu iz fizike 1999. godine.

Upisao je osnovne studije na Elektrotehničkom fakultetu, Univerziteta Beogradu, 1999. godine. Osnovne petogodišnje studije na Elektrotehničkom fakultetu završio je za nepunih 5 godina, pri čemu je svaki ispit položio u prvom ispitnom roku. Diplomirao je 2004. godine na smeru za računarsku tehniku i informatiku, sa prosečnom ocenom 9.83 i ocenom 10 na diplomskom radu kod prof. dr Veljka Milutinovića.

Završio je magistarske studije 2010. godine na Elektrotehničkom fakultetu, Univerziteta u Beogradu, smer softverski sistemi, sa prosečnom ocenom 10.00. Magistarski rad pod naslovim "Merenje i analiza softverskih sistema" je odbranio 15.9.2010. kod doc. dr Dragana Bojića. Za vreme magistarskih studija držao vežbe na predmetu programski prevodioci na Elektrotehničkom fakultetu, Univerziteta u Beogradu.

Zaposlen je u Microsoft razvojnom centru Srbija kao program menadžer na SKL Server sistemu. Pre Microsoft razvojnog centra, radio je kao projekt menadžer u softverskoj kompaniji Gowi na velikom broju projekata razvijenih za klijente u Velikoj Britaniji i Holandiji i softverski inženjer u kompaniji IPSI na projektima za klijente u Sjedinjenim američkim državama.

Čest predavač na naučnim i tehničkim konferencijama (Sinergija, TELFOR, ETRAN-a, YUINFO) i radnim grupama (radionice) u oblasti implementacije veb aplikacija pomoću Microsoft ASP.NET tehnologija i korišćenja SQL Server proizvoda. Kvalitetna predavanja na stručnim konferencijama i radnim grupama su mu 2012. godine donela MVP (engl. Most Valuable Professional) titulu za oblast razvoja veb aplikacija na Microsoft platformama, što je jedina MVP titula u oblasti razvoja veb aplikacija dodeljena na ovim prostorima.

Autor je više radova u časopisima, od čega su dva časopisa na SCI listi, kao i više radova objavljenih na domaćim i internacionalnim konferencijama u oblastima merenja softvera, procena napora, standardizacije softverskih procesa, i distribuiranih računarskih sistema.

Autor je dve knjige objavljene na srpskom jeziku: "Testiranje softvera u praksi" i "jQuery i napredne veb tehnologije". Odlukom nastavno-naučnog veća Računarskog fakulteta, knjiga "Testiranje softvera u praksi" je prihvaćena kao zvanični udžbenik na osnovnim studijama Računarskog fakulteta, za oblast Upravljanje informacijama.

Radovi objavljeni u časopisima međunarodnog značaja – M20

Radovi u međunarodnim časopisima sa SCI liste (M23):

1. Popović, J., Bojić, D.: A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle, - Computer Science and Information Systems, Vol. 9, No. 1, 2012, pp. 455-484. ISSN: 1820-0214, SCI Impact Factor: 0.549 (2012), M23.

2. Popović, J., Bojić, D., Korolija, N.: Analysis of task effort estimation accuracy based on use case point size, IET Software, Vol. 9, Issue 6, December 2015, p. 166 – 173, DOI: 10.1049/iet-sen.2014.0254 , Print ISSN 1751-8806

Zbornici skupova nacionalnog značaja – M60

Saopštenja sa skupova nacionalnog značaja štampana u celini (M63):

1. J. Popović, S. Purić, M. Batić, D. Bojić, Enhancing Use Case Point estimation method using Fuzzy algorithms, Zbornik radova sa 23. Telekomunikacionog foruma TELFOR 2015, 2015.

2. Popović J.: Implementacija procesa upravljanjima rizicima u skladu sa CMMI standardom, Zbornik radova sa 18. Telekomunikacionog foruma TELFOR 2010, Beograd, Srbija, novembar 23-25. 2010., pp. 90-93, ISBN: 978-86-7466-392-9, M63.

3. Popović J.: Implementacija procesa upravljanjima ugovorima u skladu sa CMMI standardom, Zbornik radova sa 18. Telekomunikacionog foruma TELFOR 2010, Beograd, Srbija, novembar 23-25. 2010., pp. 94-97, ISBN: 978-86-7466-392-9, M63.

4. Popović J.: Definisanje standardnih organizacionih procesa u skladu sa CMMI standardom, Zbornik radova sa 17. Telekomunikacionog foruma TELFOR 2009, Beograd, Srbija, novembar 24-26. 2009., pp. 90-93, ISBN: 978-86-7466-375-2, M63.

5. Popović J.: Implementacija procesa donošenja odluka u skladu sa CMMI standardom, Zbornik radova sa 17. Telekomunikacionog foruma TELFOR 2009, Beograd, Srbija, novembar 24-26. 2009., pp. 94-97, ISBN: 978-86-7466-375-2, M63.

6. Popović J.: Merenje tokom procesa razvoja softvera, Zbornik radova sa 53. konferencije ETRAN 2009, 15-18. Jun 2009, Vrnjačka Banja, Srbija, pp. RT4.7.1-4, ISBN: 978-86-80509-64-8, M63.

7. Popović J.: Tehnike merenja softvera, Zbornik radova XV konferencije YuInfo 2009, 8-11 mart, 2009., Kopaonik, Srbija, pp. 103-107, ISBN: 978-86-85525-04-9, M63.

8. Popović J.: Definicija procesa merenja u skladu sa CMMI standardom, Zbornik radova XV konferencije YuInfo 2009, 8-11 mart, 2009., Kopaonik, Srbija, pp. 108-112, ISBN: 978-86-85525-04-9, M63.

Magistarske i doktorske teze – M70

Odbranjena magistarska teza (M72):

1. Popović, J.: Merenje i analiza softverskih sistema, Elektrotehnički fakultet, Univerzitet u Beogradu, 2010, M72.

ПРИЛОГ 1

Изјава о ауторству

Потписани Јован Поповић

Изјављујем

да је докторска дисертација под насловом „Унапређење метода процене напора у софтверским пројектима”

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио ауторска права и користио интелектуалну својину других лица.

Потпис докторанта

У Београду, _____.

ПРИЛОГ 2

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора:	Јован Поповић
Наслов рада:	Унапређење метода процене напора у софтверским пројектима
Ментор:	др Драган Бојић, ванредни професор

Потписани Јован Поповић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао за објављивање на порталу Дигиталног репозиторијума Универзитета у Београду.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанта

У Београду, _____.

ПРИЛОГ 3

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић” да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом „Унапређење метода процене напора у софтверским пројектима”, која је моје ауторско дело.

Дисертацију са свим прилозима предао сам у електронском формату погодном за трајно архивирање. Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (*Creative Commons*) за коју сам се одлучио.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

(Кратак опис лиценци дат је на следећој страни)

Потпис докторанта

У Београду, _____.

ПРИЛОГ 3

1. Ауторство – Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство — некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство – некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство — без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.