Model zgodnjega ocenjevanja ključnih faktorjev v razvoju programske opreme

Igor Trontel

Hermanci 44, 2275 Miklavž pri Ormožu, Slovenija, igor.trontel@telekom.si

Članek obravnava model zgodnjega ocenjevanja ključnih faktorjev v razvoju programske opreme. Predstavljene so ugotovitve in povzetki raziskave ključnih faktorjev in sodobnih metod za ocenjevanje ključnih faktorjev. Predstavljeni primer ocenitve velikosti programskega proizvoda kaže, da lahko nekatere ključne faktorje ocenimo z minimalnim dodatnim trudom precej natančno že zelo zgodaj. S pomočjo te vnaprejšnje informacije lahko z večjo verjetnostjo napovemo planirane projektne cilje (kakovost, čas, stroške, funkcionalnosti) in vplivamo na ključne faktorje še preden pride do razlik med dejanskim in želenim stanjem. S pravočasno in pravilno uporabo sodobnih metod je torej možno povečati stopnjo uspešnosti projektov razvoja programske opreme.

Ključne besede: kakovost programske opreme, modeli, Cosmic FFP, SW-CMM, ocenitev virov projekta razvoja programske opreme.

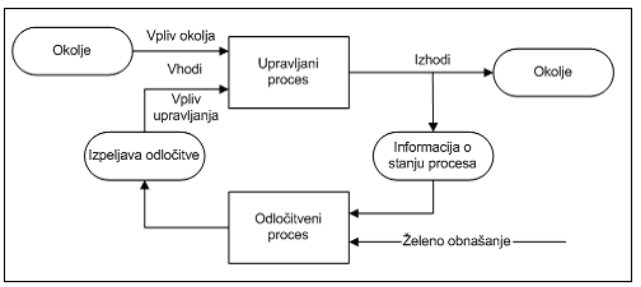
1 Uvod

Upravljanje projektov razvoja programske opreme pomeni za management velik izziv. Raziskave in aktualni primeri iz prakse kažejo, da mnogi zaidejo v težave. Standish Group[1] raziskuje vzroke za neuspehe projektov v ZDA. Na podlagi študije več kot 40.000 projektov ugotavljajo, da je stopnja uspešnosti projektov v zadnjih desetih letih dosegla 34 odstotkov. V poročilu študije (Standish, 2004) navajajo glavne razloge za povečanje stopnje uspešnosti, in sicer: projekti so manjši, v razvojnemu ciklu se uporabljajo iterativni namesto pristopi vodnega slapa, udeleženci projektov bolje razumejo in obvladujejo upravljanje projektov. Vendar je povprečje projektov, ki presežejo stroške še vedno večje od 43 odstotkov. Nadalje navajajo, da se v ZDA porabi 55 milijard dolarjev zaradi izgubljenih vrednosti ali preseženih stroškov.

Tudi svetovalna skupina Gartner je podala napoved za telekomunikacijsko panogo, in sicer da bo 75 odstotkov projektov s področja IT neuspešnih - ne bo doseglo ali planiranih rezultatov ali rokov ali pa bodo presegli planirane stroške. Najslabše je, če pride do težav na vseh področjih hkrati. Ker so danes ponudniki telekomunikacijskih storitev popolnoma odvisni od informacijske podpore, lahko neuspešen informacijski projekt povzroči motnje v njihovem poslovanju, izgubo ugleda, zmanjšanje tržnega deleža ali morda drugo nepopravljivo škodo. Nepravočasna ali manj kakovostna telekomunikacijska storitev pa lahko prepriča odjemalce, da izberejo drugega dobavitelja.

Upravljanje in vodenje projektov oz. projektni management izdelave programske opreme obsega izvajanje funkcij managementa[2] potrebnih za realizacijo vmesnih in doseganje končnega cilja. Z vidika procesnih faz managementa pa obsega upravljanje: oblikovanje informacij, odločitveni proces in izvajanje upravljanih ukrepov, kot navajajo mnogi avtorji, med drugimi J. Belak (1993).

Kot jedro uspešnega upravljanja se vedno bolj uveljavlja kombinirano odločanje na podlagi povratne (glej tudi sliko 1) in vnaprejšnje informacije. Na podlagi povratne informacije je možno kompenzirati odklone dejanskega od ciljnega stanja. Na podlagi vnaprejšnje informacije pa je možno vplivati na sistem še preden pride do odklona, oziroma je cilje možno postaviti v območje večje



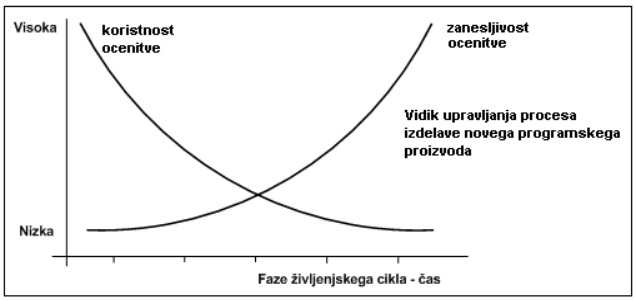
Slika 1: Shema upravljanja sistema, ki temelji na povratni informaciji (povzeto po Kljajić, 1994)

[1] Standish Group je profesionalna svetovalna skupina, ki se ukvarja z temeljnimi raziskavami v IT-ju.
[2] Paniranje, organiziranje, vodenje in kontroliranje temeljnega procesa življenjskega cikla ter potrebnih resursov.

verjetnosti uresničitve. Uspešen management mora opredeliti ključne informacijske potrebe in si potrebne informacije oziroma podatke tudi pravočasno priskrbeti.

Pri upravljanju procesa razvoja programske opreme obstaja zaradi same specifične narave programske opreme problem pridobivanja kakovostnih vnaprejšnjih informacij, saj do določene razvojne faze ni na razpolago natančnih podatkov o nekaterih ključnih faktorjih, ki vplivajo na potreben razvojni napor za realizacijo projekta oziroma na čas trajanja, kakovost in stroške projekta.

V fazi planiranja je treba izhajati iz ocen ključnih faktorjev, ki pa so lahko bolj ali manj natančne. B. Boehm, (1981 in 2000) predstavlja, da je lahko koeficient napake v oceni velikosti programskega proizvoda v začetni fazi življenjskega cikla tudi do 400 odstotkov. Napačna ocena povzroči izdelavo nerealnega plana in na podlagi tega izvajanje ter kontroliranje projektnih aktivnosti, ki so usmerjene k nerealnemu cilju. Odločitve na podlagi napačnih projekcij lahko povzročijo kritično povečanje stroškov ali trajanje projekta in s tem neželene posledice v poslovanju podjetja. Vnaprejšnja ustrezna informacija pri upravljanju procesa razvoja programske opreme je torej lahko zelo koristna. Ko se življenjski cikel nadaljuje izvemo vedno več o naravi in velikosti programskega proizvoda oziroma o drugih karakteristikah ključnih faktorjev, vendar ima pridobljena informacija vedno manjši pomen. Tako pridemo do znanega paradoksa, da koristnost informacije z izvajanjem aktivnosti življenjskega cikla razvoja eksponencialno pada, obratno pa narašča njena zanesljivost (glej sliko 2).



Slika 2: Paradoks ocenitve velikosti programskega proizvoda (Prirejeno po Meli 2000 in Kljajić 1994)

2 Ključni faktorji v razvoju programske opreme

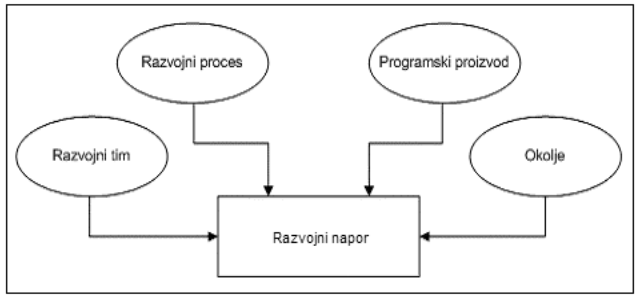
Po slovarju slovenskega knjižnega jezika pomeni faktor nekaj kar deluje, vpliva na kaj ali povzroča določeno dogajanje. Z vidika upravljanja projektov razvoja programske opreme lahko tako opredelimo ključne faktorje kot tiste, ki v največji meri vplivajo na ključne cilje projekta razvoja programske opreme s katero bomo posredno ali neposredno podprli doseganje poslovnih ciljev. Uvedba programske opreme največkrat vpliva na naslednje poslovne izide:

- povečanje funkcionalnosti,
- zmanjšanje stroškov,
- skrajšanje časa nastopa novega izdelka ali storitve na trgu in
- izboljšanje kakovosti izdelka ali storitve.

Pri raziskavi literature najdemo mnoga dela (med drugimi M. Paulk, 1997), ki opredeljujejo vplive med rezultati in zmogljivostjo razvojnega procesa, med velikostjo programskega proizvoda in časom trajanja ali stroški projekta oziroma mnoge faktorje in njihove različne vplive. Mnogi avtorji dajejo ključen pomen faktorjem, ki vplivajo na proces razvoja, drugi dajejo prednost metodologiji razvoja ali izbrani tehnologiji. Nekaterim sta najpomembnejša motiviranost in usposobljenost razvojnega tima.

Bradford Clark (1996) je opredelil naslednje štiri najpomembnejše skupine faktorjev, ki vplivajo na potreben napor pri razvoju programske opreme (glej tudi sliko 3), in sicer:

- programski proizvod,
- razvojni proces,
- razvojni tim in
- okolje.



Slika 3: Najpomembnejše skupine faktorjev, ki vplivajo na razvojni napor (povzeto po Clark 1996)

Velikost razvojnega napora je običajno podana v človek-mesecih. Na podlagi tega podatka lahko s pomočjo algoritmičnih modelov, ki slonijo na primerjavi podatkov (bench-marking), dokaj natančno predvidimo čas trajanja projekta, stroške in kakovost programske opreme. Za ocenjevanje razvojnega napora je razvito več algoritmičnih modelov.

3 Metode za ocenjevanje ključnih faktorjev

Razvoj programske opreme je zelo dinamičen, programski proizvodi pa postajajo vse večji in kompleksnejši. Eden od osnovnih pogojev za uspešno izvedbo projekta razvoja programske opreme je pravilna ocena časa trajanja in stroškov celotnega projekta oziroma posameznih faz. Na te ključne projektne izide vpliva več faktorjev. Pri tem pomeni ocenjevanje ključnih faktorjev napovedovanje njihovih vrednosti. Ocena nam mora povedati, katera vrednost je najbolj verjetna, oziroma katera vrednost je tista, ki bo v končni fazi enako verjetno nižja ali višja od dejanskega rezultata. Tako je na začetku življenjskega cikla razvoja eden izmed glavnih problemov izbrati in uporabi-

ti metodo ali skupino metod s pomočjo katerih bomo prišli do dovolj zanesljivih napovedi potrebnega časa in stroškov razvoja.

V splošnem lahko razvrstimo metode ocenjevanja v naslednje skupine:

- Ekspertne ocene.
- Od vrha navzdol (Top – down)
- Od dna proti vrhu (Bottom – up)
- Algoritmični modeli

Ekspertne ocene

Metoda temelji na izkušnjah vodje projekta oziroma eksperta iz preteklih projektov. Natančnost ocen je odvisna od izkušenj, znanja in objektivnosti ocenjevalca. Metodo lahko izboljšamo tako, da povabimo k ocenjevanju večje število ekspertov ter od njih zahtevamo več ocen: pesimistično (x), optimistično (y) in najbolj verjetno (z). Končna ocena je povprečje, ki ga izračunamo s pomočjo enačbe (x+y+4z)/6. Takšen način nam normalizira cenitev enega samega eksperta.

Na splošno temeljijo ocene ekspertov predvsem na izkušnjah in podobnostih med preteklimi projekti in projektom, ki ga ocenjujemo. V praksi pa najpogosteje najdemo projekte, ki si v posameznih elementih niso zelo podobni. Razlike lahko bistveno vplivajo na naš novi projekt. Pri tem je lahko npr. produktivnost razvijalcev zelo različna. Tudi dva razvijalca na enem modulu ne bosta končala dvakrat hitreje kot eden. Različna organizacija načrtovanja in razvoja programske opreme lahko bistveno vpliva na rezultate projekta.

Tako je ocenjevanje s pomočjo eksperta preveč izpostavljeno napakam, ki so rezultat ocenjevalčeve subjektivnosti ali pa celo napačnega razumevanja problema. Naslednja težava te metode je tudi v tem, da ne moremo oceniti natančnosti ocene, ki je podana. Tako se ta metoda največkrat uporablja za zbiranje grobih predhodnih podatkov in informacij.

Ocenjevanje od zgoraj navzdol

Pri metodi »od zgoraj navzdol« se skušajo najprej oceniti celotni stroški projekta. Nato se celoten projekt razbije na posamezne dele oziroma faze projekta. Za posamezno fazo se določijo posamezni deleži od celote. Npr. za specifikacijo bomo potrebovali 20% celotnega dela na projektu, za načrtovanje 20%, za izvedbo 30% in za testiranje 30% dela na projektu. To metodo lahko uporabljamo le v primeru izkušenj iz preteklih projektov in le če so si projekti med seboj dovolj podobni, sicer lahko ocene bistveno odstopajo od pravih vrednosti.

Ocenjevanje od spodaj navzgor

Ocenjevanje s pomočjo metode »od spodaj navzgor« je lahko natančno vendar je strokovno in časovno zahtevno. Celoten projekt se razdeli na več faz in ciljev, znotraj posamezne faze se opredelijo posamezne aktivnosti, ki morajo doseči zastavljene cilje. Najprej je treba ovrednotiti posamezno aktivnost nato se ocene posameznih komponent sestavijo v oceno celotnega projekta. Ker so posamezni elementi ocenjevanja dokaj majhni, jih je lažje pravilno ovrednotiti. Natančnost ocene projekta po tej metodi je odvisna od natančnosti posameznih ocen.

Običajno se ta metoda uporablja pri faznem ocenjevanju projekta, kjer skušamo skozi ves življenjski cikel projekta na koncu vsake faze oceniti, koliko dela nas čaka v naslednji fazi.

Algoritmični modeli

Modeli zagotavljajo algoritme kako vhodne parametre pretvoriti v čas in stroške projekta. Te metode poskušajo čim bolje oceniti statistično relacijo med odvisno spremenljivko npr. razvojni napor in ostalimi vhodnimi faktorji, kot so npr. velikost programskega proizvoda, izkušenost tima, zmogljivost razvojnega procesa, idr. Algoritmi so razviti na podlagi statističnih raziskovanj razvojnih projektov. Pri uporabi teh metod je treba ugotoviti morebitne lokalne zakonitosti relacij med spremenljivkami v lastnem razvojnem okolju. Umerjanje poteka tako, da se izračuna prilagoditveni faktor, ki bo odpravil razliko med splošno napovedjo in napovedjo prirejeno za okolje, za katerega se uporablja.

Vsaka od zgoraj predstavljenih metod ima svoje prednosti in slabosti. Prednost algoritmičnih modelov je v objektivnosti, ponovljivosti in analitičnosti, vendar je njihova težava v točnosti vhodnih podatkov in umeritvi na lokalno razvojno okolje. Na nekem projektu je možno uporabljati več metod hkrati. V času življenjskega cikla in v odvisnosti od namena jih je smiselno med seboj dopolnjevati.

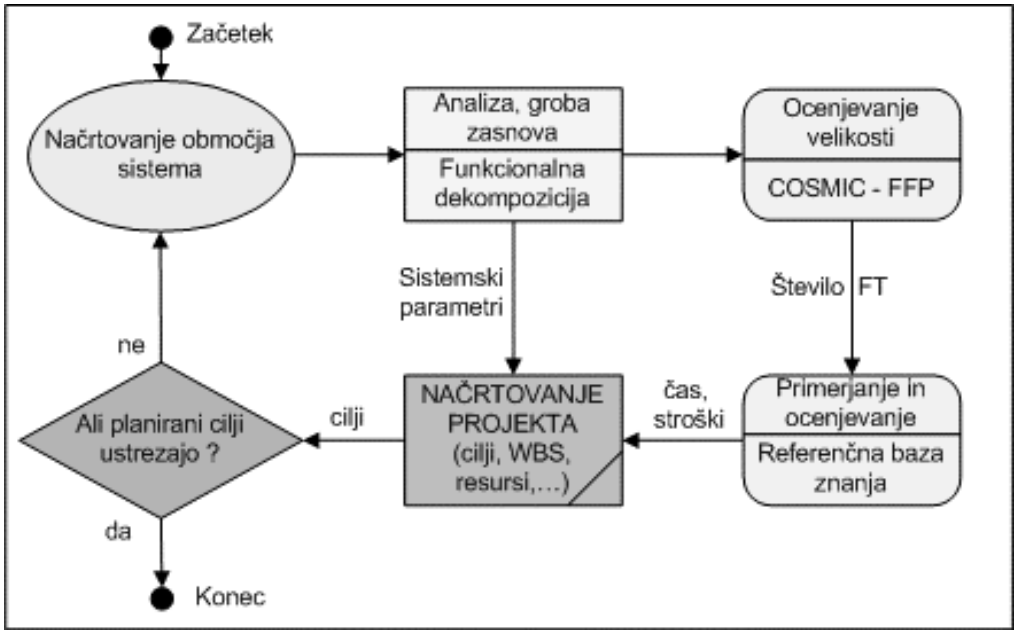
4 Model zgodnjega ocenjevanja ključnih faktorjev v razvoju programske opreme

V tem poglavju je predstavljen model uporabe dveh osrednjih metod, algoritmične in metode primerjanja[3]. Ta model je možno uporabiti v najzgodnejših fazah življenjskega cikla s ciljem pridobitve kakovostnejših vnaprejšnjih informacij, ki so nujne pri načrtovanju projekta in za odločanje o izvedbi nadaljnjih upravljalnih ukrepov.

Ob danih zahtevah po funkcionalnostih in kakovosti nove programske opreme nas zanima predvsem čas implementacije in stroški, ki jih bo projekt v času izvajanja življenjskega cikla povzročil. Ob proučevanju literature in v praksi lahko hitro ugotovimo, da je najpomembnejši ključni faktor, ki vpliva na stroške in čas trajanja razvoja, velikost programske opreme.

Model zgodnjega ocenjevanja ključnih faktorjev na sliki 4 predvideva iterativni pristop pri ocenjevanju programske opreme. V zgodnji fazi načrtovanja projekta niso znane podrobnosti o programskem proizvodu. Pomembno uporabno informacijo o območju programskega proizvoda pridobimo z analizo funkcionalnih zahtev in zasno-

[3] Metoda primerjanja je uspešna, če temelji na referencah dolgoletnega in obširnega zbiranja podatkov.



Slika 4: Model zgodnjega ocenjevanja ključnih faktorjev

vo programskega proizvoda[4]. Vedeti moramo kaj bo vključeno v programski proizvod in kaj ne, oziroma kje so meje načrtovanega proizvoda. Ta analiza zahteva zelo tesno sodelovanje razvijalcev in uporabnikov programske opreme. Analiza in zasnova programskega proizvoda mora biti usmerjena v potrebe poslovanja podjetja. Pri analizi se lahko uporabijo različne strategije. V primeru zelo kompleksnega načrtovanega območja je primerno uporabiti strategijo prototipiranja. S prototipiranjem se poveča skupno znanje razvijalcev in uporabnikov o bodočem programskem proizvodu.

Funkcionalne specifikacije, ki jih pridobimo z analizo funkcionalnih zahtev pomenijo vhodne podatke za ocenitev funkcionalne velikosti programskega proizvoda in osnovne parametre za nadaljnje načrtovanje ciljev projekta.

Za pridobitev ocene funkcionalne velikosti programskega proizvoda lahko izberemo eno izmed treh standardiziranih metod[5]. Rezultat, ki ga pridobimo s pomočjo izbrane metode je ocena količine funkcionalne velikosti programskega proizvoda, ki je podana s številom funkcijskih točk – FT.

Za nadaljnje načrtovanje rezultatov projekta je v procesu inicialnega načrtovanja treba uporabiti metodo primerjanja, in sicer na podlagi referenčnih vrednosti – baze znanja, pridobimo za dano velikost programskega proizvoda oceno velikosti stroškov in časa trajanja projekta. Načrtovane rezultate projekta torej ocenimo na podlagi pridobljenega števila funkcijskih točk in referenčnih statističnih vrednosti v bazi znanja. Če ne razpolagamo z lastno bazo znanja, je treba uporabiti zunanje vire[6]. Uporaba zunanjih virov pa je lahko zelo koristna, npr. kadar ugotavljamo učinkovitost lastnega razvojnega procesa.

Inicialna ocena rezultatov projekta pogosto preseže naročnikove pričakovane cilje - čas dobave programskega proizvoda, pričakovane stroške ali stopnjo tveganja[7] za dano vsebino in kakovost programskega proizvoda. Takrat običajno sledi zmanjševanje funkcionalnosti, ki naj bodo dobavljene ali izbira drugačne strategije razvoja programskega proizvoda. Pri pogajanju o funkcionalni velikosti novega programskega proizvoda je potrebno tesno sodelovanje razvijalcev in uporabnikov programske opreme. Dogovoriti se je treba o funkcionalnostih, ki so obvezne in pomembne ter o manj pomembnih, ki so lahko razvite v naslednjih verzijah. Sledi ponovno ocenjevanje spremenjenega območja programskega proizvoda in napovedovanje novih vrednosti porabe proračuna in časa trajanja projekta po različnih scenarijih, ki so odvisni od vključenih funkcionalnosti.

Ko je območje programskega proizvoda enkrat opredeljeno, so na podlagi števila funkcijskih točk in drugih parametrov razvoja programske opreme, s predpostavko referenčnih vrednosti, opredeljene tudi začetna ocene po-

[4] Rezultat te faze je običajno hierarhična dekompozicija najpomembnejših funkcionalnosti in druge funkcionalne specifikacije.
[5] Standardizirane metode so: IFPUG Function Point Analysis, Mark II – Function Point Analysis in COSMIC – Full Function Points.
[6] Npr.: ISBSG – International Software Benchmarking Standard Group. ISBSG zbira in vrednoti podatke, ki temeljijo na številu FT (potreben napor, stroški, čas razvoja) na projektih razvoja programske opreme iz mnogih držav: Avstralije, Avstrije, Kanade, Danske, Nemčije, Indije, Japonske, Nove Zelandije, Norveške, Poljske, Velike Britanije in ZDA.
[7] Tveganja odpovedi pri projektih prenove programske opreme, ki ima npr. 500 funkcijskih točk je manjše kot 20% in je bistveno manjše v primerjavi s tveganjem pri programski opremi, ki ima npr. 5000 funkcijskih točk, kjer je tveganje nedokončanja projekta blizu 40%. To tveganje je nesprejemljivo za marsikatero podjetje.

trebnega napora, trajanja projekta, potrebnih resursov in z njimi povezanih stroškov.

Torej že na podlagi dovolj kakovostne ocene velikosti programske opreme[8] in postavljenih začetnih ciljev projekta lahko management naročnika odloča o izbiri strategije nabave. Pri tem so na razpolago tri osrednje strategije lastnega razvoja, zunanjega razvoja ali nakupa novega programskega proizvoda in medsebojne kombinacije osrednjih strategij.

Pred dokončno odločitvijo o lastnem razvoju, je v fazi načrtovanja programskega proizvoda, smiselno nadaljevati ocenjevanje parametrov projekta in pri tem upoštevati značilnosti lokalnega razvojnega okolja.

Če se pri začetni ocenitvi ugotovi, da je planirani čas razvoja novega programskega proizvoda sicer sprejemljiv, vendar ni na razpolago dovolj notranjih resursov je odločitev o zunanjem razvoju smiselna. Pri tem je možno uporabiti funkcijske točke kot kvantitativno osnovo za pogajanje o ceni novega programskega proizvoda.

Če planirani čas trajanja razvoja programskega proizvoda presega pričakovanja naročnika ostaja strategija nakupa novega programskega proizvoda kot edina možna rešitev.

5 Primer ocenitev funkcionalne velikosti programskega proizvoda po metodi Cosmic FFP

V tem poglavju je podan povzetek poteka in rezultatov ocenitve funkcionalne velikosti konkretnega programskega proizvoda iz raziskave I. Trontel (2005). Na podlagi primera lahko ugotovimo, da je možno že zelo zgodaj z minimalnim trudom pridobiti zelo kakovostno oceno velikosti programskega proizvoda, in sicer če le dovolj natančno poznamo funkcionalne zahteve.

5.1 Programski proizvod - prototipa CNM

Najpomembnejša funkcionalnost, ki jo želimo pridobiti z razvojem programskega proizvoda CNM (Cable Network Management), je upravljanje podatkov o kabelskih kapacitetah v prostoru. V fazi analize zahtev smo se odločili za izdelavo prototipa, ki bo omogočil razvijalcem in uporabnikom boljše razumevanje vsebine obravnavanega področja ter preizkus novih IT tehnologij – integracije javanske aplikacije[9] z GIS (geografskim informacijskim sistemom), oziroma njegovimi programskimi proizvodi.

Osnovne funkcionalnosti prototipa CNM so:

- Pregled in urejanje osnovnih šifrantov,
- Pregled objektov v geografskem prostoru,
- Mikropozicioniranje objektov v geografskem prostoru.

Tip programskega proizvoda:

- Poslovna aplikacija.

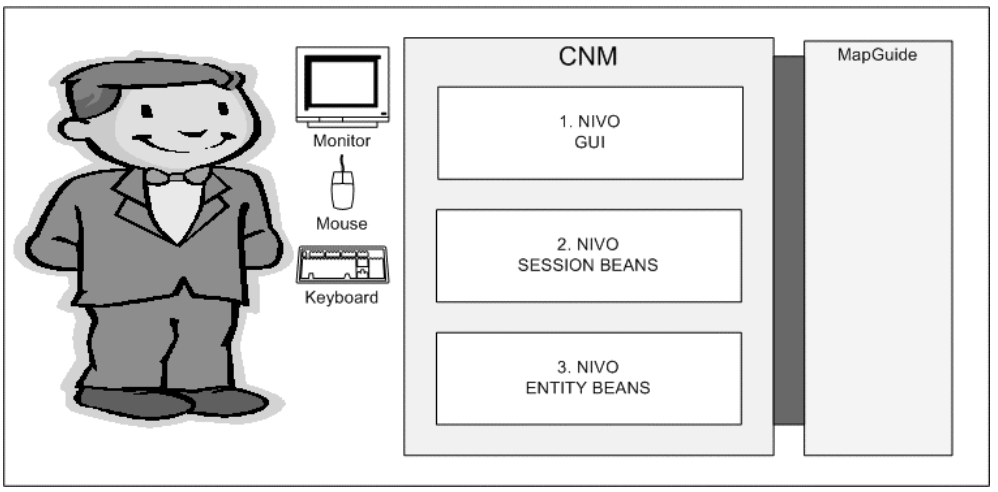
Razpoložljivi dokumenti:

- Specifikacija funkcionalnosti
- Groba zasnova programskega proizvoda.

5.2 Metoda Cosmic FFP

COSMIC-FFP metoda je danes standardizirana - ISO/IEC 19761:2003. Uporablja se za ocenjevanje funkcionalne velikosti[10] programskega proizvoda. Zasnovana je za ocenjevanje poslovne programske opreme, programske opreme za podporo sistemov v realnem času in kombinacije obeh. Ne omogoča pa ocenjevanja programske opreme, ki vsebuje kompleksne matematične algoritme ali procesno obdelavo spremenljivk zvoka in slike.

Metoda je tako kot druge iz te skupine neodvisna od uporabljenih implementacijskih odločitev. Rezultat metode je numerična vrednost količine funkcionalne velikosti programske opreme, ki predstavlja mero za funkcionalne



Slika 5: Meje prototipa programskega proizvoda CNM

[8] V petem poglavju je predstavljen primer kakovostne ocenitve funkcionalne velikosti programskega proizvoda po metodi Cosmic FFP.
[9] Aplikacije razvite s programskim jezikom java.
[10] Funkcionalna velikost – velikost dobavljenega programskega proizvoda, ki je bil določen z funkcionalnimi zahtevami naročnika.

zahteve naročnika. Izražen je v številu funkcijskih točk - FT.

5.3 Izvedba ocenitve

5.3.1 Identifikacija meje prototipa programskega proizvoda CNM

Meje prototipa programskega proizvoda CNM so opredeljene na podlagi specifikacij in grobe zasnove. CNM bo tri-nivojski programski proizvod, ki bo integriran s komercialnim programskim proizvodom MapGuide. Glej tudi sliko 5.

5.3.2 Identifikacija kandidatov obravnave

Na podlagi funkcionalnih zahtev so identificirani kandidati proženja dogodkov, funkcionalnih procesov in podatkovnih skupin.

a) Identifikacija kandidatov proženja dogodkov

Identificiran je končni uporabnik, ki bo sprožil posamezni funkcionalni proces z določenim gumbom ali izbiro ustreznega podatka.

b) Identifikacija kandidatov funkcionalnih procesov

Identificirani so funkcionalni procesi v treh glavnih skupinah:

- Pregled in urejanje šifrantov.
- Pregled objektov v geografskem prostoru.
- Mikropozicioniranje objektov v geografskem prostoru.

c) Identifikacija kandidatov podatkovnih skupin

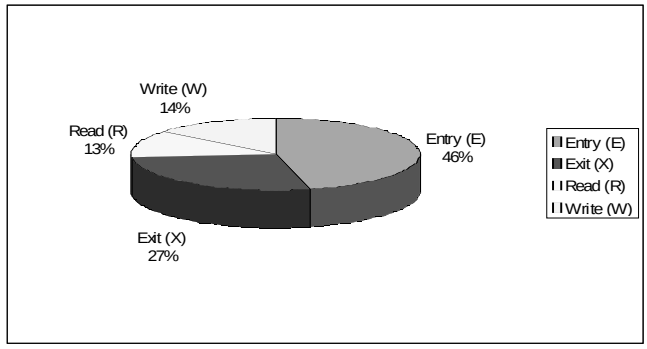
Identificirane so naslednje skupine podatkov

- Statusi - skupina statusov gumbov
- Zahteve - skupina zahtevanih akcij
- Podatki - skupina šifrantov in drugih objektov

5.3.3 Identifikacija pod - procesov in izvedba ocenitve

Na podlagi opredeljenih vlog se vsi identificirani kandidati uvrstijo v model COSMIC FFP. Vsak funkcionalni proces mora biti sprožen z dogodkom in mora biti opredeljen do posamezne podatkovne skupine. Za vsak funkcionalni proces so identificirani pod-procesi. Vsak funkcionalni premik znotraj pod-procesa[11] je določen z enim izmed štirih tipov premika: vhodni (**E**ntries), izhodni (e**X**its), preberi (**R**eads) in zapiši (**W**rites).

V prvi skupini glavnih procesov – Pregled in urejanje šifrantov je bilo ocenjeno skupno število 104 FFP. V drugi skupini glavnih procesov – Pregled objektov v prostoru je bilo ocenjeno skupno število 48 FFP. V tretji skupini glavnih procesov – Mikropozicioniranje je bilo ocenjeno skupno število 32 FFP. Če razporedimo število premikov glede na tip lahko predstavimo graf na sliki 6.



Slika 6: Graf razmerij podatkovnih premikov pod-procesov CNM

5.4 Poročilo o ocenitvi velikosti CNM

Na podlagi funkcionalnih zahtev v začetni fazi življenjskega cikla razvoja prototipa programskega proizvoda CNM je identificirano 38 funkcionalnih procesov. Skupna ocena funkcionalne velikosti je 184 CFSU oziroma funkcijskih točk (FT). Ko preračunamo FT v število vrstic programske kode - SLOC, dobimo za java2 programski jezik 8464 SLOC[12].

Poročilo po implementaciji

Po implementaciji prototipa CNM smo ugotovili njegovo dejansko velikost, in sicer 8800 SLOC. Razlika med ocenjeno vrednostjo in dejansko je minimalna, odstopanje znaša manj kot 4%.

6 Ugotovitve

Za management je torej prenova ali nabava nove programske opreme stalen izziv – kako hkrati doseči vsebino in kakovost programske opreme v želenem času in v okviru planiranih stroškov. Ugotovili smo, da so ključni faktorji tisti, ki v največji meri vplivajo na ključne cilje projekta. Pri tem ima največji vpliv prav velikost programskega proizvoda.

Za ocenjevanje ključnih faktorjev in njihovega vpliva na ključne cilje je v svetu razvitih, poznanih in uporabnih več metod od ekspertnih ocen, pristopov od vrha navzdol ali od dna proti vrhu do algoritmičnih modelov idr. Prednost algoritmičnih modelov je v objektivnosti, ponovljivosti in v analitičnosti, vendar je njihova težava v točnosti vhodnih podatkov in usmeritvi na lokalno razvojno okolje. V odvisnosti od namena in ciljev je metode smiselno uporabljati v celotnem življenjskem ciklu projekta.

V tem prispevku je predstavljen model zgodnjega ocenjevanja ključnih faktorjev v razvoju programske opreme v katerem je osrednja metoda ocenjevanje funkcionalne velikosti programskega proizvoda. Na podlagi predstavljenega primera lahko ugotovimo, da je možno

[11] Nekaj primerov: Branje statusa gumba, Branje podatkov, ...

[12] Preračun je narejen na podlagi Cost Xpert baze znanja, in sicer 1 FT predstavlja za java 2 programski jezik 46 SLOC.

zelo zgodaj z minimalnim trudom pridobiti zelo kakovostno oceno velikosti programskega proizvoda, če le dovolj natančno poznamo funkcionalne zahteve.

Ko pridobimo kakovostno oceno o velikosti programskega proizvoda, lahko s pomočjo referenčne baze znanja pridobimo tudi kakovostne vnaprejšnje informacije o dveh najpomembnejših izidih projekta – stroških in trajanju.

Tako lahko s pomočjo vnaprejšnjih informacij:

- bolje načrtujemo potrebne resurse in rezultate projektov
- z večjo verjetnostjo pričakujemo uresničitev projektov v načrtovanih okvirih
- vplivamo na ključne faktorje v razvoju programske opreme še preden pride do razlik med dejanskim in želenim stanjem

S pravočasno in pravilno uporabo sodobnih metod je torej možno povečati stopnjo uspešnosti projektov razvoja programske opreme.

Literatura

Belak, J. (1993) Podjetništvo, politika podjetja in management, Založba Obzorja, Maribor.

Boehm, B. (1981) Software engineering economic, New Jersey, Prentice Hall PTR.

Boehm, B. (2000) Software cost estimation with Cocomo II, Prentice Hall PTR, New Jersey.

Clark, B. (1996) The Effects of Software Process Maturity on Software Development Effort, Computer Science Department University of Southern California, Los Angeles.

Kljajić, M. (1994) Teorija sistemov, Založba Moderna organizacija, Kranj.

Meli, R. (2000) On the applicability of Cosmic – FFP for measuring software throughout its life cycle, Escom – Scope 2000, Munich.

Paulk, M. (1997) The Capability Maturity Model, Addison - Wesley Longman, Inc.

Trontel, I. (2005) Model ocenjevanja ključnih faktorjev v razvoju programske opreme, magistersko delo, Univerza v Mariboru, Fakulteta za organizacijske vede.

Drugi viri

Standish. (2004) Project Success Rates Improved Over 10 Years, internet (http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish), maj 2004.

Igor Trontel je leta 1998 diplomiral na Ekonomsko-poslovni fakulteti Univerze v Mariboru. Zaposlen je v Telekomu Slovenije, kjer je vodil zasnovo in projekt razvoja lastne programske opreme TKIS. TKIS je eden od treh največjih gradnikov IS v Telekomu Slovenije. Po vpeljavi v produkcijo je vodil službo za vzdrževanje in razvoj te programske opreme, od konca leta 2004 pa vodi službo za SAP. Leta 2005 je zaključil magistrski študij na Fakulteti za organizacijske vede na smeri 'Kakovost in varnost informacijskih sistemov'.