Niskobudžetni životni ciklus razvoja softvera

Stanišević Ilja, Marković Vesna, Pavlović Valentina
Visoka poslovna škola strukovnih studija VIPOS u Valjevu
Valjevo, Srbija
ilja.stanisevic@vipos.edu.rs; vesna.markovic@vipos.edu.rs;
valentina.pavlovic@vipos.edu.rs

Obradović Slobodan
Matematički institut SANU,
Beograd, Srbija
slobo.obradovic@gmail.com

Sadržaj—**Postojeći modeli životnog ciklusa razvoja softvera nisu prilagođeni primeni u niskobudžetnim uslovima. U ovom radu je opisana konstrukcija novog, niskobudžetnog modela životnog ciklusa. Istraživanje je zasnovano na ranije detektovanim karakteristikama niskobudžetnog razvoja poslovnih softverskih sistema. Konstrukcija modela se bazira na Zahmanovom okviru, kao najpotpunijem metamodelu poslovnog entiteta. U radu se predlaže novi model životnog ciklusa, koji određuje i novu metodu razvoja softvera. Konačno, prikazana je i evaluacija predloženog rešenja koja pokazuje da je ovako konstruisan metod bolje prilagođen niskobudžetnim uslovima od postojećih metoda.**

Ključne reči: metode za projektovanje i razvoj softvera; niskobudžetni uslovi; Zahmanov okvir; evaluacija primenljivosti metoda projektovanja softvera, UML

I. UVOD

Kao posledica složenosti procesa razvoja softverskih sistema, došlo je do razvoja većeg broja modela životnog ciklusa. Izbor adekvatnog modela u velikoj meri određuje metodu i strategiju razvoja, te može značajno unaprediti efikasnost procesa u odnosu na željene ciljeve. Izbor prave metode nije lagan zadatak, budući da on zavisi od tipa projekta, osobina modela, ali i od ciljeva koji se pri realizaciji projekta žele postići. Tako, kao primarni cilj metode može biti postavljen kvalitet proizvoda, ali i minimizacija rizika, brzina projektovanja i razvoja, rigorozna kontrola razvojnog procesa itd.

Životni ciklus je put od inicijacije do terminiranja projekta razvoja softvera. Podela u faze omogućava menadžerima da kontrolišu i upravljaju aktivnostima na disciplinovan, sređen i metodičan način, te da tako adekvatno odgovore na komplikovane zahteve za promenama u toku razvojnog procesa [1]. Upotreba koncepta životnog ciklusa omogućava podelu složenih zadataka na manje i jednostavnije, lakšu identifikaciju zadataka, obezbeđivanje zajedničke terminologije, lakšu kontrolu finansijskih resursa projekta, sistematičnu evaluaciju opcija i alternativa, integraciju aktivnosti, upravljanje neizvesnošću itd [2]. Životni ciklus, u najvećoj meri, determiniše i metod i strategiju razvoja softverskog sistema.

Danas postoji veći broj modela životnog ciklusa razvoja softvera. U zavisnosti od njihove fleksibilnosti, najčešće ih delimo u tri osnovne kategorije.

Ovaj rad je realizovan u okviru projekta III 44006 Ministarstva obrazovanja i nauke Republike Srbije)

U prvu kategoriju spadaju teški modeli, tj. modeli bazirani na sekvencijalnom pristupu. Primeri ovakvih modela su kaskadni model [3], V-model i W-model. Ovi modeli posmatraju projekat kao jedan monolitni ciklus. Detaljna specifikacija zahteva dozvoljava direktnu kontrolu procesa [4]. Ovi modeli su procesno orjentisani i vođeni planom, te u sebe uključuju detaljlno planiranje i kodifikaciju procesa pomoću kojih razvoj treba da bude efikasna i predvidljiva aktivnost [5].

Teški modeli su najbolje prilagođeni situacijama u kojima su zahtevi i tehnologija dobro poznati i kad na raspolaganju imamo dovoljno resursa. Osnovni nedostatak predstavlja nepostojanje povratne sprege u procesu, kao i nedovoljna fleksibilnost modela.

Drugu kategoriju čine modeli bazirani na inkrementalnom pristupu, tj. modeli životnog ciklusa srednje težine. Oni teže uspostavljanju balansa između formalizma teških modela i fleksibilnosti agilnih modela. Osnovna njihova karakteristika je podela projekta na seriju povezanih podprojekata, tzv. inkremenata. Svaki od njih stvara dodatnu vrednost u vidu novih mogućnosti i funkcionalnosti [6]. Na ovim principima su razvijene mnoge metode za brz razvoj aplikacija (*eng. Rapid Application Development – RAD*). One obično u sebi uključuju razvoj prototipova, korišćenje sofisticiranih softverskih alata i male, ali dobro obučene i visoko specijalizovane razvojne timove.

Inkrementalni pristup je posebno koristan u slučajevima kada nije moguće unapred dati specifikaciju zahteva [1], tj. kad ni razvojni tim ni korisnik nisu u stanju da precizno definišu domen problema. Takođe je veoma pogodan kada postoji potreba za brzim razvojem ili kada se funkcionalni zahtevi često menjaju.

Treću kategoriju čine metode bazirane na lakom modelu životnog ciklusa, tj. agilne metode. One su bazirane na osnovna četiri agilna principa [7] koji naglašavaju kreativnost pojedinaca i orjentisani su ka što bržem ostvarenju krajnjeg proizvoda, funkcionalnog softvera. Na ovim principima se razvio veći broj metoda projektovanja softvera. Najrašireniје su ekstremno programiranje – XP [8], skram [9], metode kristal [10], razvoj na bazi karakteristika – FDD i metod dinamičkog razvoja sistema – DSDM.

Agilne metode su korisne u stalno promenljivim okruženjima, gde postoji potreba za parcijalnim rešenjima,

pružaju podršku konkurentnom razvoju i permanentnoj integraciji u okruženje kupca [1].

II. NISKOBUDŽETNI USLOVI RAZVOJA SOFTVERA

Sve ove metode razvoja softvera su nastale u tehnološki visoko razvijenim sredinama koje raspoloažu dovoljnim resursima za razvoj složenih softverskih sistema. To je razlog zbog kojeg ni jedna metoda u sebi nema inkorporiranu minimizaciju potrebe za resursima kao njen osnovni strateški cilj.

S druge strane, uslovi razvoja softvera u društvima u tehnološkom razvoju su značajno razlikuju od onih koji vladaju u informatički razvijenim društvima, te, stoga, postojeće metode nisu adekvatne za primenu u niskobudžetnim sredinama [11]. Značajan nedostatak, pre svega, finansijskih resursa, kao posledicu ima drastičnu restrikciju tehničkih (kako softverskih, tako i hardverskih) resursa. Detektovana je redukcija i kadrovskih resursa. Kadrovska redukcija se odnosi kako na kvalitet, tako i na raspoloživi kvantitet angažovanih kadrova, i to i sa korisničke strane, i sa strane razvojnog tima. Konačno, nestabilno društveno okruženje tipično za društva u razvoju generišu i socijalne restrikcije, koje svoj izvor imaju kako unutar razvojnog tima, tako i izvan njega, u širem društvenom kontekstu. Polazeći od svih ovih detektovanih ograničenja i efekata koji oni imaju na razvoj poslovnih softverskih sistema u niskobudžetnom okruženju, došlo se do osobina koje bi metoda za projektovanje i razvoj softverskih sistema trebalo da poseduje.

Niskobudžetna metoda razvoja softverskih sistema treba da, pored ostalog, omogući višestruki angažman multifunkcionalnog osoblja organizovanih u male i fleksibilne timove čiji članovi imaju različite nivoe stručne osposobljenosti. Metoda treba da omogući parcijalna i modularna rešenja, pri čemu prvenstvo realizacije imaju za korisnika najvažniji i najkritičniji moduli. Ovakva realizacija omogućava faznu realizaciju i implementaciju u zavisnosti od raspoloživih sredstava korisnika. Metoda mora da bude fleksibilna i da omogućava jednostavne i efikasne modifikacije. Aktivnosti koje se odnose na planiranje i kontrolu treba da budu minimizirane (ali ne i izostavljene), dok dokumentacija ne mora biti sveobuhvatna, ali treba da bude jasna i pregledna [12]. Njena osnovna funkcija u niskobudžetnom okruženju više nije da puži prikaz celokupnog procesa projektovanja i realizacije, te implementacione detalje sistema, već služi da olakša komunikaciju kako sa korisnikom, tako i između članova razvojnog tima, te da pruži nformacije potrebne za brzu i efikasniju naknadnu modifikaciju sistema. Sam sistem treba da bude jednostavan i pouzdan, ali ne da podržava sve aspekte korisničkog poslovanja, već one koji su kritični i za korisnika najznačajniji.

III. ZAHMANOV OKVIR

Prilikom dizajniranja metode razvoja softverskog sistema, treba voditi računa o uključenosti svih aspekata u proces razvoja modela. Budući da se ovde radi prvenstveno o poslovnim softverskim sistemima, bilo je potrebno usvojiti metamodel organizacionog entiteta koji će osigurati sveobuhvatnost rešenja. Za tu svrhu je izabran Zahmanov okvir, kao najsveobuhvatniji i danas najšire prihvaćen metod prikaza arhitekture preduzeća.

Zahmanov okvir je struktuiran i formalan način za predstavljanje celokupne arhitekture preduzeća. On predstavlja presek dve istorijske klasifikacije. Prvu dimenziju predstavljaju primitive komunikacije i može se opisati kao odgovor na fundamentalna pitanja: šta, kako, kada, ko, gde i zašto. Integracija odgovora na ova pitanja pruža potpun prikaz kompleksnih ideja. Odgovori na ova pitanja daju informacije o: podacima, funkcijama, lokacijama, akterima, vremenu i motivaciji. Druga dimenzija je derivirana iz procesa transformacije apstraktne ideje u instancu, i njene vrednosti su u Zahmanovom okviru označene kao identifikacija, definicija, reprezentacija, specifikacija konfiguracija i instancijacija [13].

Usled svoje dvodimenzionalne prirode, Zahmanov okvir se predstavlja kao dvodimenzionalna matrica, kao što prikazuje Tabela I.

TABELA I. TABELARNI PRIKAZ ZAHMANOVOG OKVIRA

ZAHMANOV OKVIR	**ŠTA** (podaci)	**KAKO** (funkcije)	**GDE** (lokacija)	**KO** (akteri)	**KADA** (vreme)	**ZAŠTO** (motivacija)	
PLANER (kontekstualni)	spisak bitnih stvari	spisak procesa	spisak lokacija	spisak organizacija	spisak događaja	spisak poslovnih ciljeva	**FI**
VLASNIK (konceptualni)	model poslovnih entiteta	model poslovnih procesa	logistička mreža	organizacioni dijagram	model poslovnih događaja	model poslovne strategije	
DIZAJNER (logički)	normalizovani model podataka	arhitektura aplikacije	arhitektura mreže	athitektura ljudske interakcije	dijagram događaja	model poslovnih pravila	
TEHNOLOG (fizički)	fizički model podataka	dizajn sistema	tehnologija mreže	arhitektura prezentacije	dijagram kontrole toka	dizajn poslovnih pravila	**FP**
IZVOĐAČ (vankontekstni)	fizička specifikacija podataka	specifikacija programskih komponenti	specifikacija mrežnih komponenti	specifikacija komponenti interfejsa	specifikacija događaja	specifikacija poslovnih pravila	**FRI**
RADNIK (proizvodni)	aktuelni podaci	izvršni program	komunikacione mogućnosti	obučeni korisnici sistema	poslovni događaji	primenjena pravila	**FE**

LEGENDA: **FI – faza inicijacije** **FRI – faza realizacije i integracije**
FP – faza projektovanja **FE – faza eksploatacije**

Druga dimenzija se vezuje za perspektivu posmatrača, tj. za ulogu onoga koji daje odgovore na pitanja iz prve dimenzije, ili za delokrug iz kojeg se odgovori daju. Shodno tome, vrednosti druge dimenzije Zahmanovog okvira su često zamenjene nazivom uloge posmatrača koji odgovara fazi transformacionog procesa, te su tako definisane perspektive: planera (kontekstualni delokrug), vlasnika (konceptualni delokrug), dizajnera (logički model), tehnologa (fizički model), izvođača (vankontekstni delokrug) i izvršnog radnika (proizvodni delokrug).

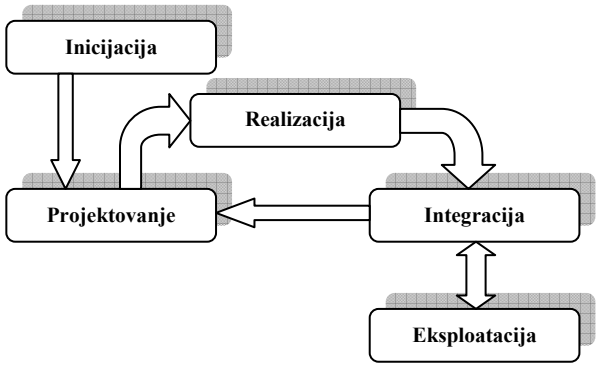
Svaka ćelija matrice se dalje posebno razlaže i razrađuje, pri čemu se ne definiše nivo detaljnosti do kojeg se ide. Ovakav model se može dalje, po potrebi, proširiti, uvođenjem novih dimenzija (npr. sigurnosti, tehnologije i sl.), te se tada matrica transformiše u višedimenzionalnu kocku [14].

Ne postoji pravilo do kog nivoa detaljnosti se ide, već to zavisi od značaja koji dati problem iz određene perspektive za korisnika ima. Ovako definisana matrica pokriva sve aspekte objekta koji se njome predstavlja.

Zahmanov okvir predstavlja ontološku strukturu i metamodel, a ne metodologiju, te se njime ne impicira ništa o načinima implementacije, detaljnosti modela, korišćenim alatma i primenjenim metodama [15]. Nedefinisanjem ovih bitnih aspekata, postignuta je značajna fleksibilnost Zahmanovog okvira. Takođe, ovakav pristup čini Zahmanov okvir izuzetno pogodnim za primenu prilikom konstrukcije drugih metoda i modela.

IV. KONSTRUKCIJA ŽIVOTNOG CIKLUSA

Da bi model životnog ciklusa bio sveobuhvatan, on treba da inkorporira sve vrednosti Zahmanovog okvira po obe dimenzije (tj. treba da uključi sve ćelije Zahmanove tabele). S druge strane, ograničenost resursa koja karakteriše niskobudžetne uslove zahteva krajnje minimiziranje aktivnosti. Stoga je kriterij korišćen pri kreiranju modela životnog ciklusa razvoja poslovnog softverskog sistema izvođenje samo aktivnosti neophodnih za realizaciju sistema (karakteristično za agilne metode), pri čemu ne dolazi do preskakanja i zanemarivanja određenih aspekata realizacije (do kakvog često dolazi pri implementaciji agilnih metoda [16]). Ova strategija nas je dovela do modela koji je prikazan na Sl.1.



Slika 1. Model životnog ciklusa niskobudžetnog razvoja softvera

Ceo niskobudžetni projekat započinje fazom inicijacije koja daje konceptualni okvir projekta i definiše domen problema. Iterativni deo životnog ciklusa se sastoji od naizmeničnih koraka projektovanja, realizacije i integracije sistema u realno okruženje korisnika. Ne postoji ograničenje broja iteracija, ali ni limitiranje vremenskog trajanja pojedine iteracije. Ovo je posebno bitno u niskobudžetnom okruženju, jer omogućava da se projekat privremeno obustavi, dok se ne steknu uslovi (tj. obezbede potrebni resursi) za njegov nastavak. Za vreme trajanja ovakvih prekida, nema prepreka da se operativno koristi deo sistema koji je realizovan i integrisan u sklopu prethodno okončanih iteracija. Završna faza je faza eksploatacije, ali ona nije i terminirajuća faza projekta. Moguće je iz faze eksploatacije ponovo ući u iterativni deo životnog ciklusa radi naknadne modifikacije ili proširenja sistema.

Ovako definisan životni ciklus razvoja softvera obuhvata sve aspekte (tj. u svojim koracima inkorporira sve ćelije) Zahmanovog okvira. Time smo obezbedili sveobuhvatnost na ovaj način kreiranog modela.

V. FAZA INICIJACIJE

U početnoj fazi, fazi inicijacije, realizuju se sve aktivnosti na kontekstualnom i konceptualnom nivou, kao i definisanje arhitekture, aktera i vremena na logičkom nivou. Ovaj korak treba da definiše domen i opšti okvir razvojnog projekta na visokom nivou apstrakcije.

Osnovni rezultat inicijalne faze predstavlja okvirni inicijalni plan. Da bi obuhvatio sve konceptualne i kontekstualne aspekte projekta u skladu sa Zahmanovim okvirom, okvirni inicijalni plan treba da sadrži sledeće elemente:

- spisak poslovnih ciljeva i opis poslovne strategije organizacije korisnika;
- konceptualni dijagram relevantnih poslovnih entiteta i njihovih odnosa;
- skup modela osnovnih poslovnih procesa;
- dijagram prostorne, organizacione i interpersonalne distribucije sistema (na logičkom nivou);
- model (ciklus) poslovnih događaja;
- model arhitekture aplikacije (na logičkom nivou).

U ovom radu se ne razmatra konkretna implementacija. Generalna strategija izrade okvirnog inicijalnog plana predviđa prilagođavanje implementacije potrebama razvojnog tima. Kao veoma pogodni za realizaciju plana, kao jedno moguće rešenje, su se pokazali UML dijagrami [17], koji su, s jedne strane, dovoljno opšti da bi obuhvatili sve aspekte okvirnog inicijalnog plana, a, s druge strane, moguće ih je relativno brzo generisati raznim postojećim softverskim alatima (čak i upotrebom običnog MS Worda, npr.), te njihova izrada ne zahteva značajniju potrošnju razvojnih resursa.

VI. FAZA PROJEKTOVANJA

U fazi projektovanja se vrši klasično projektovanje i dizajniranje sistema. Generiše se model podataka na logičkom i fizičkom nivou, kreira se arhitektura aplikacije i dizajn sistema,

definiše se tehnologija mreže, definiše se dijagram kontrole toka, te se modeliraju i dizajniraju poslovna pravila korisnika.

U niskobudžetnim uslovima, ipak, postoji jedna bitna razlika. Ne vrši se projektovanje celokupnog sistema, već se i sam projekat generiše modularno, pri čemu nije obavezno projektovanje svih modula, već samo onih koji će se u prvoj iteraciji realizovati. Takav dizajn mora biti otvoren i mora biti takav da omogućava naknadni dalji razvoj.

Bitna osobina svakog niskobudžetnog projekta je zahtev za maksimalnom fleksibilnošću, o čemu posebno treba voditi računa u ovoj fazi. Prednost treba uvek davati rešenjima koja omogućavaju parametarsko podešavanje realizacije poslovnih pravila, tj. treba maksimalno moguće izbegavati projektna rešenja kod kojih se pravila ugrađuju u kod sistema.

Kao izlaz iz faze projektovanja imamo relacione dijagrame baze podataka, dijagrame osnovnih klasa sa osobinama i deklaracijom metoda, dijagrame toka aktivnosti, kao i prikaz poslovnih pravila korisnika. Pri izradi ove dokumentacije se treba skoncentrisati samo na one vidove dokumentacije koji će biti korišćeni u realizaciji i modifikaciji sistema, dok čisto dokumentacione i kontrolne aspekte treba minimizirati ili, čak, u slučaju manjih projekata i razvojnih timova, potpuno zanemariti.

VII. Faza realizacije i integracije

Prilikom realizacije samog sistema, treba koristiti sve metode i tehnike programiranja koje ne vode značajnom direktnom povećanju zahteva za resursima. Ovde se posebno misli na upotrebu skupih programerskih alata koji zahtevaju dodatnu specijalističku obuku angažovanog ljudstva (karakteristično za RAD metode). Treba i do krajnosti racionalizovati angažman skupog, visoko obučenog kadra, te iskoristiti manje stručan kadar za generisanje standardnih delova koda, kao što su čeone komponente aplikacija. Takođe, izrada kodnih šablona i projektnih obrazaca u velikoj meri može smanjiti potrošnju resursa prilikom realizacije.

Paralelno sa realizacijom se odvija i faza integracije. Optimalno je kada je moguće razvoj realizovati u okruženju kupca. Ukoliko takvi uslovi ne postoje, treba vršiti što češću integraciju koda u realno eksploataciono okruženje. Ovakva praksa omogućava rano uočavanje i eliminaciju funkcionalnih grešaka, što smanjuje potrebu za resursima. Takođe, permanentna integracija omogućava lakše prihvatanje sistema od strane korisnika.

Kao proizvod faze realizacije i integracije imamo potpuno operativni kod sistema. Treba napomenuti da, iako potpuno funkcionalan, ovaj sistem ne mora da sadrži i sve predviđene funkcionalnosti. Modularna i fazna realizacija podrazumeva da se delovi sistema mogu realizovati naknadno, pri čemu to ne umanjuje funkcionalnost već realizovanih modula sistema.

VIII. Faza eksploatacije

Fazu eksploatacije karakteriše operativni programski sistem koji se potpuno koristi u realnom radnom okruženju kupca.

Za razliku od klasičnih, niskobudžetni projekti se ne završavaju ulaskom u fazu eksploatacije. Niskobudžetno okruženje se odlikuje velikom dinamičnošću, tako da je samo pitanje vremena kada će se nastaviti rad na sistemu, bilo kroz modifikaciju postojećih, bilo kroz nadogradnju novih funkcionalnosti. Stoga možemo zaključiti da se niskobudžetni projekat razvoja poslovnog softvera nikada i ne okončava.

IX. Evaluacija rešenja

Za evaluaciju ovako definisanog životnog ciklusa razvoja softvera je korišćen dvofazni evaluacioni metod baziran na ekspertskoj evaluaciji, koji se zasniva na modifikovanom Vasilekasovom obrascu [18]. Ovaj metod daje evaluaciju primenljivosti metode projektovanja i realizacije softvera u odnosu na zahtevani cilj. Ciljevi mogu biti različiti, zavisno od razvojnog okruženja. Tako možemo imati cilj minimizacije vremena izrade, minimizaciju broja angažovanog osoblja, minimiziranje rizika izrade, maksimiziranje kvaliteta sotvera, maksimiziranje kontrole razvojnog procesa i slično. U ovom istraživanju je kao cilj nametnuto minimiziranje potrebe za resursima.

Sama evaluaciona metoda se sastoji od dve faze evaluacije, pri čemu se u prvoj fazi determinišu osobine koje se ocenjuju i određuje njihova važnost u odnosu na cilj koji se želi postići u razvoju softverskog sistema. U drugom koraku se vrši ekspertska evaluacija prethodno određenih osobina.

Za proveru dobijenih rezultata, koristi se i kontrolna funkcija koja je, u skladu sa principom istraživačke triangulacije, bazirana na drugačijim pretpostavkama. Ova funkcija se zasniva na metodologiji izbora najbolje ponude koju koristi Svetska banka [19], na način kako je to izloženo u radu [20].

Završni obrazac na opisani način izvedene evaluacije je dat jednačinom (1).

$$PM = \frac{1}{N} \sum_{i=osobina1}^{osobinaN} k_i * \frac{\sum_{j=1}^{m_i} vo_{ij}}{m_i} \qquad (1)$$

Pri čemu su nam:

PM – vrednost prilagođenosti metode datim uslovima;
N – broj ocenjivanih osobina;
$\mathbf{k_i}$ – ocena važnosti i-te osobine;
$\mathbf{m_i}$ – ukupan broj eksperata koji procenjuju i-tu osobinu;
$\mathbf{vo_{ij}}$ – ocena za i-tu osobinu dodeljena od j-tog eksperta.

Ocena važnosti izabrane osobine k_i pri nametnutom scenariju je definisana petostepenom skalom kao:

1.-nebitna osobina;
2.-osobina ograničenog značaja;
3.-značajna osobina;
4.-veoma značajna osobina;
5.-kritično značajna osobina.

Pored ocene važnosti osobine, eksperti vrše i ocenjivanje vrednosti same osobine. Kako su izabrane osobine različite po svojoj prirodi i, u opštem slučaju, moguće ih je meriti na najraznovrsnije, međusobno neuporedive načine, izabrano je da se ocenjivanje vrši univerzalnom petostepenom metrikom definisanom na sledeći način:

1.-ne zadovoljava zahteve;
2.-delimično zadovoljava zahteve;
3.-zadovoljava zahteve;
4.-delimično prevazilazi zahteve;
5.-značajno prevazilazi zahteve.

Treba napomenuti da se, kao rezultat, dobija mera prilagođenosti metode razvoja softvera nametnutom scenariju, tj. željenom ciljnom kriterijumu, pri čemu veći ostvaren rezultat predstavlja bolju prilagođenost metode zahtevanim uslovima razvoja. U sklopu ovog istraživanja, ciljni kriterijum predstavlja minimizacija utroška resursa, tj. metoda sa najboljim rezultatom je najpodesnija za primenu u niskobudžetnim uslovima.

Budući da je moguće da, iz bilo kog razloga, kod evaluatora prevlada subjektivnost, uvodi se i kontrolna funkcija koja predstavlja indikator ispravnosti realizovane evaluacije. Kontrolna funkcija se dobija kao rezultat obrasca datog u jednačini (2).

$$RP = \frac{C_{min}}{C} T + \frac{Q}{Q_{max}} (1 - T) \qquad (2)$$

Ovde su nam:

RP – realizovana vrednost ponude;
C – procenjena cena ponude (gubitak);
C_{min} – najmanja od svih procenjenih cena ponude;
Q – ukupan kvalitet ponude (dobitak);
Q_{max} – vrednost kvaliteta najkvalitetnije ponude;
T – težinski faktor cene u intervalu $0 < T < 1$.

Kvalitet, tj. dobitak ostvaren primenom ispitivane metode u datim uslovima određuje se na osnovu pet osnovnih oblasti u skladu sa ISO 9126 standardom, a to su funkcionalnost, pouzdanost, upotrebljivost, efikasnost i održivost, pri čemu se svaka ocenjuje ocenom od 1 do 10, gde najbolji rezultat odgovara oceni 10.

Uloga ovako definisane kontrolne funkcije nije da obezbedi apsolutan pokazatelj ispravnosti evaluacije, već da ukaže na moguće greške u rezonovanju. Ukoliko dođe do razmimoilaženja rezultata ostvarenih evaluacijom i primenom kontrolne funkcije, to ukazuje da postoji određeni uzrok neusaglašenosti kojeg treba detektovati. Ukoliko se ispostavi da se radi bilo o svesnoj ili nesvesnoj subjektivnosti evaluatora, postupak evaluacije treba preispitati i ponoviti.

Veličina C (cena – gubitak) predstavlja utrošak najbitnijeg, kritičnog resursa. S obzirom da kod primene u niskobudžetnim uslovima najbitniji resurs predstavljaju sami resursi, cena je računata kao zbir procene utroška kadrovskih, hardverskih, softverskih, vremenskih i finansijskih resursa.

Prilikom vršenja evaluacije, angažovani su eksperti koji su osim u tri do sada realizovana projekta niskobudžetnog razvoja softverskih sistema, učestvovali i u mnogim drugim projektima kod kojih su primenjivani drugačiji modeli životnog ciklusa. Za evaluaciju su izabrani najkarakterističniji modeli razvoja, i to: kaskadni model (tj. model vodopada), model brzog razvoja aplikacija (RAD), Majkrosoftov okvir za rešenja (MSF), ekstremno programiranje (XP) i ovde izloženi niskobudžetni model razvoja poslovnih sistema. Evaluacija ostalih modela može poslužiti i kao probni uzorak.

Postignuti rezultati, kako evaluacionog postupka, tako i vrednosti kontrolne funkcije, prikazani su u Tabeli II.

TABELA II. TABELARNI PRIKAZ REZULTATA EVALUACIJE RAZLIČITIH ŽIVOTNIH CIKLUSA

Ispitivana metoda	Evaluacija	Kontrolna funkcija
kaskadna	8,4419	0,5283
RAD	9,9302	0,6186
MSF	8,3488	0,5286
XP	11,7440	0,7650
niskobudžetni razvoj	12,3950	0,8700

Iz ovde iskazanih rezultata vidimo da su evaluacioni rezultati i rezultati do kojih se došlo primenom kontrolne funkcije u saglasnosti. Takođe, primećujemo da su se kao najnepodesnije pokazale glomazne, rigidne metode, kao što su kaskadna i Majkrosoftov okvir za rešenja – MSF. Ovo je bilo predvidivo, budući da obe metode u svoj fokus stavljaju kontrolu razvojnog procesa, kao i da su poznate po visokoj zahtevnosti za razvojnim resursima. Značajno bolji rezultat (za preko 30%) je ostvaren pri evaluaciji agilne metode ekstremnog programiranja. I ovo se može smatrati kao očekivani rezultat, budući da je XP poznat kao fleksibilna metoda striktno fokusirana isključivo na generisanje završnog proizvoda, koja često zanemaruje ostale aspekte razvoja.

I u postupku evaluacije, i primenom kontrolne funkcije, najviši rezultat je ostvario ovde predstavljen niskobudžetni model životnog ciklusa razvoja softvera. Na osnovu ovoga možemo zaključiti da je u ovom radu predstavljen model najbolje primenljiv u niskobudžetnim uslovima.

X. ZAKLJUČAK

U ovom radu je izložen jedan mogući pristup rešavanju problema projektovanja i realizacije softverskih sistema u niskobudžetnim uslovima. Dat je ovirni prikaz životnog ciklusa metode razvoja softvera koji u svom fokusu ima minimiziranje potrebe za resursima. Generisani životni ciklus je sveobuhvatan, budući da u svojoj konstrukciji ima impliciran Zahmanov okvir, koji predstavlja potpuni model organizacionog entiteta.

Takođe, pokazano je da je ovakav pristup bolje prilagođen primeni u niskobudžetnim uslovima od dosadašnjih rešenja. Pri tome se nije težilo da izloženi pristup bude i univerzalno optimalan. Takođe, težište rada je bio na osnovnim principima i strategiji pristupa, a ne na određivanju konkretnih postupaka i metoda.

U budućim istraživanjima je potrebno ispitati optimalnu formu realizovane projektne dokumentacije. U dosadašnjem

radu je, uglavnom, primenjivan ujedinjeni jezik za modeliranje – UML. Ipak, potrebno je detaljnije istražiti da li je ovaj pristup dovoljan, kao i da li je, sa stanovišta potrebe za resursima, i optimalan.

Takođe je potrebno istražiti primenljivost konkretnih pojedinačnih metoda, kako projektantskih, tako i programerskih, te u skladu sa rezultatima tih istraživanja, izvršiti dodatnu formalizaciju ovde definisanih koraka životnog ciklusa.

U svakom slučaju, kreiranje metode projektovanja i razvoja poslovnih softverskih sistema u niskobudžetnim uslovima u značajnoj meri bi olakšala i ubrzala informatizaciju tehnološki nedovoljno razvijenih sredina, te su, stoga, opravdana dalja istraživanja ovog problema.

LITERATURA

[1] Benediktsson O., Dalcher D., Thorbergsson, "Comparation of software development life cycles: a mjultiproject experiment", *IEE Proceedings – Software*, Vol.153, No.3, pp. 87-101, June 2006.

[2] Dalcher, D. "Life cycle design and management" in Stevens M. (Ed.) "*Project managment pathways: a practitioner's guide*", APM Press, High Wycombe, 2002.

[3] Royce, W.W. "Managing the development of large software systems: Concepts and techniques." *Proceedings of WESCON*, August 1970: VOL XIV, pg. A1-A9

[4] Boehm, B.W. "Get ready for agile methodologies with care" *IEEE Software*, No. 14, Vol. 4, pp. 64-71, 2002.

[5] Guntamukkala, V., Wen, H.J., Tarn, J.M., "An empirical study of selecting software development life cycle models", *Human Systems Managment*, 25, IOS Press, pp. 265-278, 2006.

[6] Laraman, C., Basili, V.C., "Iterative and incremental development: a brief history", *Computer*, No.36, Vol.6, pp. 47-56, 2003.

[7] Agile Alliance: *Principles of the Agile Alliance.* on-line, 2001. http://www.agilealliance.org/..

[8] Beck, K. "*Extreme Programming Explained: Embrace Change",* Boston: Addison-Wesley, 2000.

[9] Schwaber, K., Beedle, M. "*Agile Software Development with Scrum",* Upper Saddle River, NJ: Prentice Hall, 2002.

[10] Cockburn, A. "*Crystal Clear: A Human-Powered Methodology for Small Teams",* Reading, MA: Addison Wasley, 2005.

[11] Stanišević I., Pavlović V., Petrović Đ. „Komparativna analiza metoda projektovanja IS u domaćim uslovima", *Proc. of 52nd ETRAN Conference*, RT 4.3 1-4, Palić, June 08-12, 2008.

[12] Stanišević I., Obradović S. „*Characteristics of the Optimal Method for Software Design in a Low-budget Environment*", *Megatrend Review*, Vol 8, No.2, UDC 005.8:004.41, ISSN 1820-4570, pp. 597-524, 2011.

[13] Zachman, J.A., *The ZF: The Official Concise Definition of Zachman International Enterprise Architecture*. [on-line] Zachman International Inc., http://www.zachmaninternational.com/index.php/home-article, 2009.

[14] Jovanović V., Mrdalj S., Gardiner A., „*A Zachman Cube", Issues in Information Systems*, Vol VII, No. 2, pp. 257-262, 2006.

[15] Zachman, J.A., „*A Framework for Information Systems Architecture", IBM Systems Journal*, vol.26, no 3, IBM Publication G321-5298, 1987.

[16] Stanišević I., Petrović Đ., Pavlović V. "Komparativna analiza agilnih metoda projektovanja IS u našem okruženju", *INFOTEH-Jahorina* Vol. 8, Ref. E-III-16, p. 589-593, March 2009.

[17] Booch G., Rumbaugh J., Jacobson I. „*The Unified Modeling Language - User Guide",* Boston, MA, Adison Wesley Longman Inc., 1999.

[18] Vasilecas, O., Saulis, A., Dereškevičius, S. "Evaluation of information systems procurement: goal and task-driven approaches." *Information Technology and Control , Vol.35, No.3,* pp. 229-234, 2006.

[19] The World Bank, "*Standard Bidding Documents. Supply and Installation of Information Systems Two-Stage Bidding",* 2003. [on-line] http://www.worldbank.org/ itprocurementforum

[20] Stanišević I., Obradović S. „*Selecting Software Development Life Cycle Models*", International Scientific Conference UNITECH 2011, ISSN 1313-230X, pp.I393-I397, Gabrovo, Bulgaria, 18 – 19 November 2011

ABSTRACT

Existing software development life cycle models are not suitable for low-budget conditions. In this paper construction of the low-budget life cycle model is described. The investigation is developed from low-budget software development characteristics. The construction of the model itself is based on Zachman framework. As a result, the new life cycle model and software development method is proposed. Finally, evaluation of the model is provided. The evaluation shows that such derived method is better suitable for low-budget conditions than the existing ones.

LOW-BUDGET SOFTWARE DEVELOPMENT LIFE CYCLE

Stanišević Ilja, Obradović Slobodan, Marković Vesna, Pavlović Valentina