Техничко решење

Назив техничког решења

Софтверска апликација за руковање захтевима корисника која омогућује унапређење послова одржавања софтвера у малом софтверском предузећу

Аутори решења

Миша Варга, *YuTeam Software OD Зрењанин*

Горан Николић, *YuTeam Software OD Зрењанин*

Жељко Стојанов, *Универзитет у Новом Саду, Технички факултет "Михајло Пупин" Зрењанин*

Александар Жарков, *YuTeam Software OD Зрењанин*

Врста техничког решења

Софтвер (уз доказ) M85

Наручилац решења

YuTeam Software OD Зрењанин

Корисник решења

YuTeam Software OD Зрењанин

Година израде решења

2011–2012.

Област технике на коју се техничко решење односи

Софтверско инжењерство, одржавање софтвера

Решење прихваћено од

1. YuTeam Software OD Зрењанин
2. Наставно-научно веће Техничког факултета "Михајло Пупин" Зрењанин, Универзитет у Новом Саду

Начин верификације решења

Преглед техничке документације решења и тестирање решења у радном окружењу предузећа YuTeam Software OD Зрењанин

Начин коришћења резултата

Решење се користи у свакодневном раду у предузећу YuTeam Software OD Зрењанин за управљање захтевима корисника.

Техничко решење је развијено у оквиру пројекта "**Развој софтверских алата за анализу и побољшање пословних процеса**", број пројекта ТР–32044, који је финансиран од стране Министарства просвете и науке Републике Србије.

Увод

Техничко решење, софтверска апликација за руковање захтевима корисника, је развијена као интерна апликација која се користи у малпм софтверском предузећу YuTeam Software OD из Зрењанина. Софтверска апликација је у употреби од 2008 године. У оквиру пројекта *Развој софтверских алата за анализу и побољшање пословних процеса* (број пројекта ТР–32044, финансиран од стране Министарства просвете и науке Републике Србије) је започет процес анализе и унапређења процеса одржавања софтвера у предузећу. Један од циљева тог пројекта је анализа постојећег стања процеса одржавања софтвера (и софтверског решења које то подржава), уочавање недостатака и дефинисање унапређења процеса. Софтвер обезбеђује руковање захтевима корисника, пословима који прате захтева корисника и административне активности везане за процес одржавања (радни налози и фактурисање завршених послова).

Анализом процеса одржавања софтвера је утврђено да је потребно извршити модификацију репозиторијума захтева корисника, а самим тим и софтверске апликације, како би се обезбедила боља подршка за праћење појединих фаза у процесу одржава, праћење оптерећења програмера (утрошак радних сати), и праћење утицаја статуса корисника на процесирање захтева.

Овим техничким решењем је обухваћена модификација репозиторијума захтева корисника и интерне софтверске апликације која обезбеђује руковање захтевима. У другом поглављу је представљен модификован репозиторијум захтева, са нагласком на сегменте који су модификовани. У трећем поглављу је представљена софтверска апликација која ообезбеђује руковање захтевима, и то верзија која се свакодневно користи након увођења модификација.

Област на коју се техничко решење односи

Техничко решење представљено у овом документу спада у област одржавања софтвера, па ће у овој секцији бити представљени основни појмови и проблеми који постоје у тој области. Посебна пажња је посвећена проблемима које сусрећу мала софтверска предузећа, а који утичу на начин организације пословања тих предузећа, и на начин унапређења инжењерске праксе у самим предузећима.

Одржавање софтвера се у литератури посматра као последња фаза у животном циклусу софтвера којој се не посвећује довољно пажње у поређењу са развојем софтвера. Одржавање софтвера се генерално дефинише као било какав рад на софтверском систему након што он пређе у фазу употребе, чиме се на неки начин поставља јасна граница између фаза развоја и одржавања. Одржавање софтвера је управљано догађајима (*event-driven*) (Kitchenham et al., 1999). Догађаји који иницирају процес одржавања најчешће потичу од корисника (клијената, или у ширем смислу тржишта), али могу потицати и од људи који су ангажовани у развоју и одржавању софтвера.

Истраживања указују да се највећи део трошкова за софтверске системе јавља након првог издавања, тј. у фази одржавања (Lientz et al., 1978)(Sommerville, 2001)(Schach and Tomer, 2000)(Abran et al. 2004)(Jones, 2010). У литератури се могу пронаћи подаци да су трошкови одржавања између 40\% и 90\% од укупних трошкова софтвера у целом животном циклусу (Kajko-Mattsson et al., 2001). Трошкови одржавања за системе који су веома дуго у употреби вишеструко превазилазе трошкове развоја (Sommerville, 2001). Због тога је унапређење процесе одржавања софтвера веома важно за смањење трошкова, унапређење квалитета и брзине одговора на потребе корисника.

Мала софтверска предузећа чине велики удео у софтверској индустрији широм света. Највећи број тих предузећа има мање од 10 запослених који су најчешће ангажовани на текућим дневним пословима. На пример, Laporte et al. (2006) указују да у Европи 85% предузећа у сектору информациних технологија има између 1 и 10 запослених. Публиковани су многи модели за процену и унапређење процеса (нпр. CMM/CMMI, ISO 9001 Quality Management, ISO/IEC 12207, ISO/IEC 15504), али они нису прихваћени у малим софтверскиим предузећима (Richardson and von Wangenheim, 2007)(von Wangenheim et al., 2006)(Coleman and O'Connor, 2008)(Laporte et al., 2008). Разлог за то је недостатак ресурса и времена за имплементирање модела добре праксе.

Мала софтверска предузећа најчешће прилагођавају процесе и технологије својим могућностима и потребама. На то указују резултати бројних истраживања у областима као што су прихватање агилних метода (Taylor et al., 2008), употреба алата за моделовање (Thorn and Gustafsson, 2008), или управљања процесом промене софтвера у фази одржавања (Stojanov et al., 2011; Stojanov, 2012). Без обзира на предмет истраживања, за праксу у малим софтверским предузећима од пресудног је значаја компетентност запослених, као и промене у тиму које уносе додатни ризик у раду педузећа (Saastamoinen and Tukiainen, 2004). Такође, не постоји експлицитна и свеобухватна методологија одржавања софтвера са фокусом на мале софтверске организације, иако су Pino et al. (2012) дали препоруку за примену агилног приступа одржавању кроз примену методолошког оквира названог Agile_MANTEMA. Модел зрелости одржавања софтвера (*Software Maintenance Maturity Model*) презентован од стране April et al. (2005) такође није прихваћен од стране малих софтверских организација с обзиром да прописује превише детаља које мале организације не могу да имплементирају.

Имајући у виду наведена ограничења малих софтверских организација, и значај одржавања софтвера, као циљ истраживања је постављен развој модела репозиторијума одржавања који је *искројен* за потребе софтверског предузећа YuTeam Software OD из Зрењанина. У овом документу је описано техничко решење које се односи на унапређење процеса одржавања софтвера кроз развој модел репозиторијума корисничких захтева који обезбеђује ефикасно праћење критичних фаза у процесу, оптерећења програмера, и утицаја који статус корисника има на процесирање захтева за одржавањем. Такође, подаци из репозиторијума треба омогуће дефинисање предиктивних модела одржавања софтвера који би могли бити примењивани за процену оптерећења (*maintenance effort*) програмера (Jorgensen, 1995)(Yu, 2006) и трошкова одржавања (Sommerville, 2001) за мала софтверска предузећа.

Опис проблема који се решава техничким решењем

Проблеми оптимизације утрошка времена и ангажованости програмера током одржавања софтвера су идентификовани као есенцијални за смањење трошкова одржавања софтвера. Решавању ових проблема се може приступити на два начина:

- Пројектовањем софтвера тако да се олакшају послови одржавања. Овај приступ подразумева структурирање софтвера тако да се што једноставније могу идентификовати делови софтвера на којима је потребно интервенисати на основу захтева корисника.
- Пројектовањем софтверског решења које омогућује ефикасно праћење захтева корисника са циљем да се сагледају критичне тачке у процесу одржавања и оптерећење сваког од програмер. Овај приступ подразумева развој концептуалног модела репозиторијума захтева који обезбеђује идентификацију свих кључних тренутака у процесу, утрошак радних сати и ангажовање сваког од програмера у предузећу.

С обзиром да је ово решење развијено као део пројекта анализе и унапређења процеса у малом софтверском предузећу које је присутно на тржишту више од 10 година, одабран је други приступ у оптимизацији трошкова одржавања софтвера. Прецизније, предузеће има више од 100 клијената у Србији који користе више од 30 софтверских апликација. Приликом иницијализације процеса унапређења праксе одржавања ове софтверске апликације су већ увелико у интензивној употреби, што потврђује велики број захтева корисника. Анализа праксе одржавања је урађена за период од маја 2010 године до новембра 2011 (19 месеци). Анализом је утврђено да су од укупно 2257 захтева корисника 1901 захтева су били захтеви за одржавањем (84%), док се 356 захтева односило на остале активности (16%). Овај однос потврђује наводе из литературе који указују да је одржавање софтвера најскупљи и најзахтевнији део животног циклуса софтвера (Boehm and Basili, 2001).

Параметар који је такође неопходно узети у обзир приликом пројектовања репозиторијума је постојање уговора о одржавању између предузећа и клијената (*Service Level Agreement*, *SLA*). Важно је напоменути да предузеће има потписане уговоре о одржавању са око 30 клијената. Ови уговори поред финансијских детаља треба да обезбеде договорени ниво услуга према потребама корисника (Bouman et al., 1999)(April et al., 2000).

Овакво стање је утицало на избор приступа унапређењу праксе који се базира на развоју модела репозиторијума захтева који омогућује праћење свих трошкова одржавања софтвера. Поред репозиторијума, решење обухвата и веб базирану софтверску апликацију за праћење захтева корисника коју интерно користе програмери у предузећу. Међутим, кључни део овог решења је модел репозиторијума захтева корисника који обезбеђује праћење свих критичних времена у процесу захтева, расподелу оптерећења програмера током одржавања софтвера, и процену утицаја статуса клијента на процес одржавања.

Модел репозиторијума захтева треба да обезбеди:

- Праћење времена тријаже захтева. То је време које прође од тренутка прихватања захтева до тренутка када је неко од програмера прихватио захтев за реализацију.
- Праћење времена реализације захтева. То је време које прође од тренутка када је програмер прихватио захтев, па до тренутка када је завршио све послове у вези захтева.
- Праћење утрошка радних сати за решавање захтева да би се добило реално оптерећење програмера.
- Идентификацију апликација које су најкритичније за одржавање.
- Процену утицја статуса корисника (приоритет корисника) на процес одржавања.

Стање решености проблема у свету: Приказ и анализа постојећих решења

Управљање процесом одржавање софтвера је од фундаменталног значаја за софтверску индустрију. У пракси се одржавање посматра као скуп активности, или услуга које софтверска организација пружа својим клијентима. Услуге се најчешће наплаћују кокрисницима на бази утрошеног времена и ангажовања програмера на пословима који су неопходни за реализацију услуге (Bhatt et al., 2004). Квалитативном анализом процеса

промене софтвера у фази одржавања Briand et al. (1994) су утврдили да је за ефикасан процес одржавања неопходно постојање репозиторијума захтева који прати историју свих активности (historical database).

Репозиторијуми који постоје у оквиру одржавања користе за планирање и процену трошкова одржавања. Ови репозиторијуми садрже податке који су релевантни за откривање трендова и релација које су битне током животног циклуса софтвера (Kagdi et al., 2007). У литератури која је фокусирана на одржавање софтвера репозиторијуми се најшешће користе за процену релевантности појединих активности, и за анализу процеса тријаже захтева или извештаја о дефектима. Највећи број истраживања је базиран на репозиторијумима који се односе на пројекте отвореног кода због доступности тих репозиторијума. Такве студије користе архиве за рад са верзијама (*Concurrent Versions System*) или исистеме за праћење дефеката (*BUGZILLA*).

На пример, у (Kagdi et al., 2011) је приказан приступ додељивању одговорности за имплементацију захтева (отклањање дефеката или захтева за новим функционалностима) програмерима са одговарајућом експертизом у одржавању, а који се базира на информацијама које се издвајају из репозиторијума. Приступ је тестиран на подацима који су прикупљени из репозиторијума за три пројекта отвореног кода ArgoUML, Eclipse и Koffice. Овај приступ указује да је у репозиторијуму неопходно водити евиденцију о томе који су програмери стекли искуство и експертизу у решавању одређених проблема у одржавању софтвера.

Kajko-Mattsson (1999) је приказала процес промене софтвера у оквиру одржавања софтвера у оквиру велике софтверске организације ABB у Шведској. Анализом процеса се могу уочити два основна типа захтева у одржавању у оквиру организације: захтев за модификацијом и захтев за реализацијом осталих активности одржавања. За оба типа захтева су постоје екранске форме за евидентирање захтева. Ове форме омогућују инжењерима да евидентирају све активности у процесу одржавања, чиме се обезбеђује да су сви подаци о одржавању софтвера доступни у централизованом репозиторијуму одржавања.

April (2010) је приказао процес унапређења праксе одржавања софтвера у софтверском предузећу Integratik из Канаде који је реализован у 2009. години. Унапређење обухвата имплементацију процеса праћења захтева. Сваки захтев корисника се евидентира, диспечира и прати формално. Активности одржавања софтвера у предузећу су анализиране и стандардизоване да би цео процес био потпуно јасан свима који су укључени у процес одржавања. Сваком захтеву се додељује радни захтев и проселеђује члану тима за одржавање. Софтверска апликација која обезбеђује подршку за праћење захтева корисника омогућује дневно праћење послова који се реализују у склопу обраде сваког захтева. Овакав приступ подразумева систематско паћење података у току процеса одржавања софтвера. На основу прикупљених података могуће је урадити процену процеса одржавања и сагледати трендове који се јављају у процесу одржавања. На основу уочених трендова је могуће даљње унапређење праксе.

Junio et al. (2011) су приказали процес руковања захтевима за одржавањем који захтеве групише у софтверске пројекте. Овај процес садржи три основне фазе: регистровање захтева, груписање захтева, и реализацију. Приступ је имплементиран у ИТ сектору PUC Minas једног од највећих бразилских универзитета. Анализа је базирана на 2088 захтева за одржавањем, а укључује 9011 радних сати у оквиру одржавања софтвера. Анализа је базирана на репозиторијуму захтева који омогућује ефикасно праћење утрошка времена у процесирању захтева. Времена која су укључена у анализу су: време распоређивања, време планирања, време сервисирања, време анализе, време имплементације, време валидације и време испоруке софтвера након завршетка свих активности. Иако у раду није имплицитно приказан сам репозиторијум захтева за одржавањем, приказани процес одржавања (*SMRMG*, *Software Maintenance Request Model Graph*), и анализа захтева за одржавањем базирана на кључним временима у процесу пружају увид у начин реализације репозиторијума захтева за одржавањем. На бази анализе података из репозиторијума, аутори су приказли приступ за унапређење процеса одржавања софтвера, и указали на добити које доноси анализа репозиторијума са захтевима за одржавањем.

Детаљни опис техничког решења

Опис техничког решења садржи опис модела репозиторијума који се користи за евидентирање свих послова који се реализују у предузећу, а односе се на захтеве корисника. Иницијална верзија овог репозиторијума са интерном веб базираном апликацијом је у употреби од 2008. године.

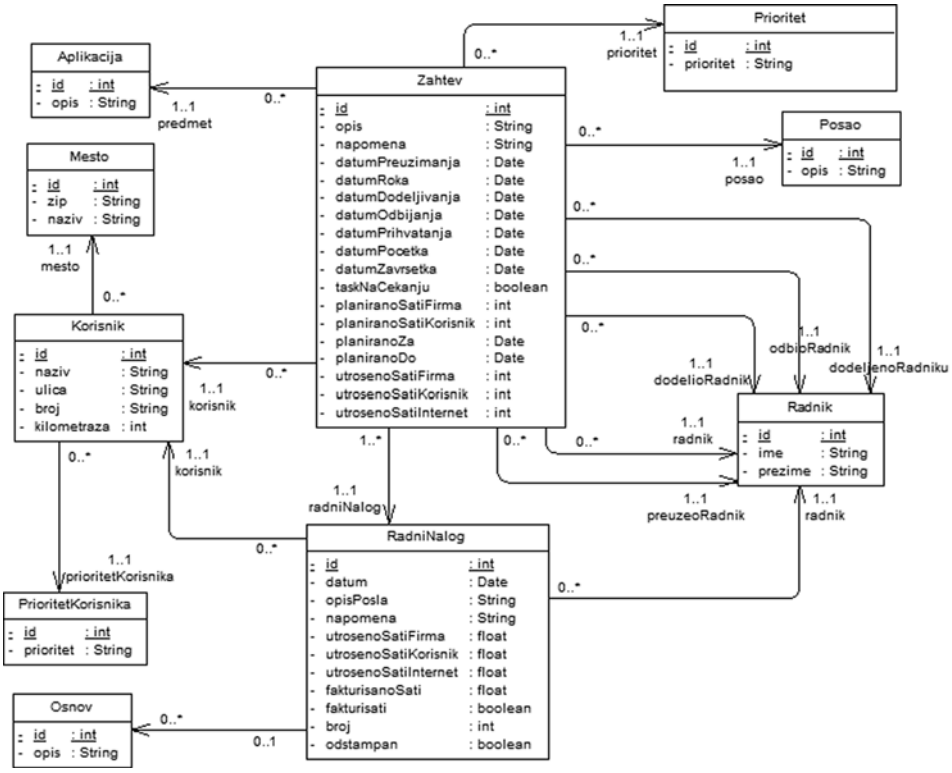
Статистичком анализом садржаја репозиторијума за период од маја 2010 до новембра 2011, и кроз радне сесије (*feedback meeting*, (Dyba et al., 2004)) у предузећу на којима је дискутовано текуће стање су уочени недостаци у процесирању захтева корисника. На основу анализе прикупљених статистичких података (квантитатина анализа), и забележених дискусија (квалитативна анализа) су дефинисани правци побољшања процеса захтева корисника. Један од тих праваца је и унапређење модела репозиторијума захтева и интерне апликације. Унапређење реозиторијума захтева се односи на:

- Прецизно дефинисање кључних времена у процесу да би се могло пратити процесирање захтева са јасним увидом у трајање појединих фаза у процесу. Овим се постиже и детаљнији увид у утошак времена и у оптерећење програмера.

- Дефинисање приоритета корисника да би се обезбедило ефикасније процесирања захтева корисника који су значајнији за пословање самог предузећа. Дефинисање приоритета корисника је јасно мотивисано финансијским ефектима пружања услуга корисницима.

Модел репозиторијума са захтевима корисника

Репозиторијум захтева за одржавањем треба да обезбеди чување и ефикасну употребу свих захтева за одржавањем софтвера. То значи да је потребно дефинисати модел који омогућује да се прате сви подаци који су релевантни за процес одржавања софтвера. Модел репозиторијума треба да обухвати и техничке и организационе аспекте одржавања софтвера чиме се обезбеђује свеобухватнији увид у целокупан процес одржавања софтвера. Такође модел репозиторијума треба да омогући анализу прикупљених података са циљем да се омогући оптимизација и унапређење процеса одржавања. Модел репозиторијума захтева за одржавањем је приказан на слици 1. Структура репозиторијума је дефинисана дијаграмом класа који се користи у оквиру објектно оријсентисаног моделовања применом обједињеног језика моделовања (Booch et al., 1999).



Слика 1. Модел репозиторијума захтева за одржавањем софтвера

Према моделу приказаном на слици 1, два основна ентитета су **Zahtev** и **RadniNalog**, па ће бити детаљније образложени у наставку.

Захтев корисника

Захтев корисника је ентитет који обезбеђује евидентирање података релевантних за проблем самог захтева, али и за процесирање захтева. Сам захтев клијента је опсан следећим пољима:

- *Напомена* (napomena). Ово је текстуално поље у које програмер који реализује захтев уноси напомене у вези захтева, реализације, или финасијске детаље. На пример ако је реализација захтева трејала 10 минута (грешка која се једноставно отклања) напомена може бити да није потребно фактурисати овај захтев.

- *Посао на чекању* (taskNaCekanju). Ово је поље које је логичког типа, и омогућује програмеру да одложи посао на реализацији захтева до даљњег, а да то буде видљиво и осталим програмерима. Одлагање се врши ако је програмер превише заузет и не може да довољно брзо реализује захтев, или ако је корисник који је пријавио захтев на *црној листи*} (видети секцију *Приоритет корисника*).

- *Планирани сати у фирми* (planiranoSatiFirma). Број планираних сати за реализацију захтева које ће радник утрошити у предузећу.

- *Планирани сати код корисника* (planiranoSatiKorisnik). Број планираних сати за реализацију захтева које ће радник утрошити код корисника.

- Датум и време за када радник планира да ради на захтеву (planiranoZa).

- *Датум и време до када радник планира да ради на захтеву* (planiranoDo). Времена планирања су потребна због интерне организације рада у предузећу. На основу увида у већ планиране обавезе програмер може да планира рад на новим захтевима.
- Број стварно утрошених сати у предузећу (utrosenoSatiFirma).
- Број стварно утрошених сати код корисника (utrosenoSatiKorisnik).
- Број стварно утрошених сати на интернету (utrosenoSatiInternet).

Утрошени сати на реализацији захтева се евидентирају на нивоу половине радног сата, а служе за евиденцију стварног утрошка времена на реализацији захтева. Тиме со добија увид у стварно оптерећење програмера, а са друге стране евидентирани утрошени сати служе као основа за креирање радног налога и наплату извршеног посла.

Захтев такође садржи поља која обезбеђују да се прате учесници у процесу реализације сваког захтева. Тиме се добија увид у ангажовање програмера на решавању захтева корисника. То су следећа поља:

- Радник који је примио захтев од корисника и евидентирао га у репозиторијуму (асоцијација preuzeoRadnik).
- Радник који је доделио захтев некоме од програмер на реализацију (асоцијација dodelioRadnik).
- Радник који је одбио захтев прослеђен од другог радника у прдузећу (асоцијација odbioRadnik). За сваки захтев је могуће да се деси више покушаја додељивања и одбијања захтева, али се у репозиторијуму памти само последњи.
- Радник коме је додељен захтев за реализацију (асоцијација dodeljenoRadniku).
- Радник који реализује захтев (асоцијација radnik).

Додатне информације које се односе на сам захтев, као што су приоритет захтева, корисник који је пријавио захтев, софтверска пликација на коју се односи захтев и опис посла који се реализује су имплементиране у независним ентитетима који су асоцијацијама повезани са ентитетом захтева.

Кључни допринос овог техничког решења је јасно дефиисање појединих времена која се појављују у процесирању захтева, а обезбеђују праћење статуса захтева. С обзиром на важност ових информација оне су детаљно описане у засебној секцији *Процесирање захтева корисника*.

Радни налог

Радни налог је ентитет који садржи податке који су битни за евидентирање посла придруженог захтеву корисника, као и наплату урађеног посла. Део података као што су опис и напомена се преузима из ентитета корисник, док се остали подаци уносе независно од ентитета захтев. На пример, радни налог треба да обезбеди да се кориснику достави опис посла и тачна евиденција утрошка радних сати на основу чега се доставља фактура. Иницијално се број радних сати преузима из ентитета захтева, али се овде може ручно променити ако се процени да би био проблем да се фактурише стваран број радних сати. На овакве одлуке битно утиче пословна политика предузећа, као и однос који је успостављен са корисником чији се захтев посматра. Датум фактурисања је датум када се завршавају сви послови који се односе на захтев корисника, и технички и административни. То је датум када се затвара радни налог.

Приоритет корисника

Појам приоритета корисника је уведен са циљем да се изврши класификација корисника која би омогућила приоритизацију њихових захтева. Приоритет корисника се поставља када се уносе подаци о њему (најчешће је то предузеће или организација) у репозиторијум, а базира се на субјективном мишљењу руководства предузећа о успостављеном пословном односу са корисником или на постојању уговора о одржавању. Подаци о кориснику се могу мењати у зависности од нивоа успостављене сарадње. Када се прими захтев од корисника, прво се провери његов приоритет и након тога приоритет захтева који је корисник предложио. На тај начин се обезбеђује класификација захтева на два нивоа:

- Први ниво класификације је на основу интерне класификације корисника која је доступна свим запосленима у предузећу.
- Други ниво класификације је на основу приоритета који је корисник предложио за сам захтев, а који може бити висок, средњи и низак (ентитет Prioritet). Овај приоритет може променити програмер који је прихватио захтев за реализацију на основу сагледавања проблема или текућих обавеза.

Приоритет корисника је у моделу репозиторијума представљен класом PrioritetKorisnika, а тренутне вредности приоритета корисника су према нивоу:

- *Висок* (Корисник са уговором о одржавању). У ову категорију спадају корисници који имају потписан уговор о одржавању софтвера на месечном нивоу, и њихови захтеви се увек прво разматрају и решавају.
- *Средњи* (Корисник који редовно плаћа услуге). Ово су корисници који немају уговор о одржавању, али имају редовне захтеве за одржавањем. Ови корисници редовно плаћају услуге одржавања.
- *Низак* (Корисник са повременим одржавањем). То су корисници који имају повремене захтеве, а своје финансијске обавезе најчешће не испуњавају благовремено.
- *Црна листа*. Ово су корисници који нису своје обавезе, пре свега финансијске, испуљавали већ дужи временски период. Реализација ових захтева се одлажа (за захтев се поставља у репозиторијуму вредност taskNaCekanju). Врло често се ови захтеви у опште не реализују.

Процесирање захтева корисника

Стање или статус било ког захтева се може одредити на основу прћења релевантних датума (овде појам датум обухвата и датум и време) у току процесирања захтева. Ови датуми су доступни у репозиторијуму захтева, а утврђени су током пројектовања концептуалног модела репозиторијума захтева корисника. Модел захтева садржи датуме који описују специфичне *тачке* у процесу захтева. Међутим, анализа и процењивање трајања појединих фаза у процесу подразумева израчунавање трајања тих фаза на основу датума који су евидентирани у репозиторијуму. Кључни датуми у процесу захтева су:

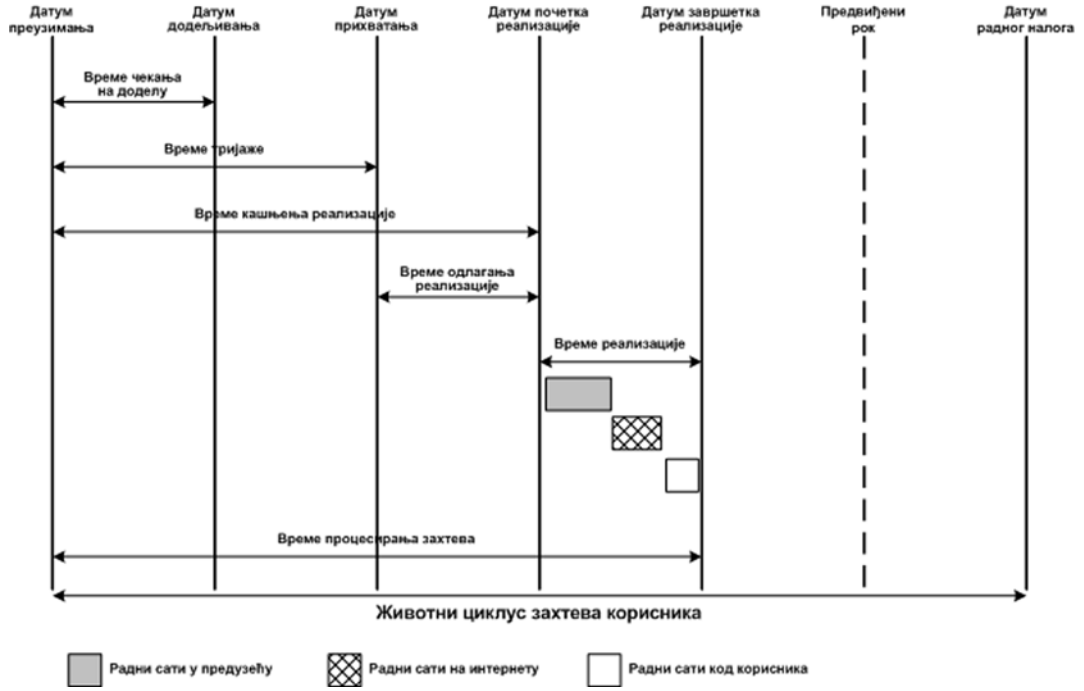
- *Датум преузимања захтева* (Zahtev::datumPreuzimanja). Датум када је неко од запослених у предузећу примио захтев и евидентирао га у репозиторијум захтева.
- *Рок за реализацију захтева* (Zahtev::datumRoka). Датом када корисник очекује да његов захтев буде реализован. Овај датум се поставља у репозиторијуму приликом евидентирања захтева.
- *Датум додељивања захтева* (Zahtev::datumDodeljivanja). Датум када програмер шаље захтев другом програмеру на реализацију. Овај датум је у највећем броју случајева исти са датумом преузимања пошто се додела захтева врши одмах након преузимања. У многим случајевима, програер захтев додељује самом себи.
- *Датум одбијања захтева* (Zahtev::datumOdbijanja). Датум када је програмер одбио захтев који му је прослеђен пошто он није задужен за одржавање софтверске апликације на коју се односи захтев. Овај датум у највећем броју случајева није евидентиран у репозиторијуму пошто се захтеви најчешће исправно додељују. Ако програмер одбије захтев, мора га проследити другом програмеру. Ако постоји више додељивања и одбијања захтева, у репозиторијуму ће бити евидентирани само последње додељивање и одбијање.
- *Датум прихватања захтева* (Zahtev::datumPrihvatanja). Датум када је неко од програмера прихватио захтев за реализацију (асоцијација zahtevRadnik). Захтев може бити примљен од другог запосленог у предузећу, или га програмер може доделити самом себи.
- *Датум почетка реализације захтева* (Zahtev::datumPocetka). Датум када је програмер почео да реализује захтев. Овај датум у највећем броју случајева није исти са датумом прихватања захтева, и зависи од осталих обавеза програмера у тренутку прихватања захтева.
- Датум завршетка реализације захтева (Zahtev::datumZavrsetka). Датум када програмер завршава све техничке послове у вези захтева и то евидентира у репозиторијуму. Датум завршетка одговара датуму затварања посла придруженог захтеву (асоцијација zahtevPosao).
- *Датум радног налога* (RadniNalog::datum). Датум када је програмер комплетирао све организационе активности у вези захтева и евидентирао радни налог у репозиторијуму. Радни налог се штампа и заједно са фактуром доставља клијенту. Овај датум означава крај целог животног циклуса захтева, а обухвата завршетак и техничких и организационих активности.

Процесирање захтева корисника са означеним датумима из репозиторијума и кључним временским интервалима је приказано на слици 2. Са циљем да се временско процесирање захтева скрати у предузећу је свакој апликацији додељено неколико програмера који могу прихватити захтев корисника. У животном циклусу захтева корисника се могу уочити следеће фазе:

- *Време чекања на доделу*. Време које прође од пријема захтева до додељивања захтева програмеру. У овој фази је захтев само евидентиран у репозиторијуму.
- *Време тријаже*. Време које прође од пријема захтева па док га неко од програмера не прихвати за реализацију. Ово време битно утиче на кашњење у обради захтева корисника. У неким случајевима корисници пријављују захтев програмеру за којег знају да је одговоран за софтверску апликацију на коју се односи захтев, и тада програмер који је прихватио захтев га додељује себи и одмах га прихвата.

С обзиром да у предузећу постоји тачно дефинисана расподела одговорности програмера за поједине апликације, програмер који прима захтев врло брзо може да пронађе групу одговорних програмера и да захтев проследи некоме од њих. Време тријаже ће бити дуже ако је програмер који је примио захтев веома заузет и проследи га другом програмеру који је такође одговоран за софтверску апликацију на којој треба интервенисати.

- *Време кашњења реализације*. Време које прође од пријема захтева до почетка реализације послова на захтеву. На ово време утичу тријажа захтева, кашњење због одлагања доделе захтева и одлагање реализације због оптерећености програмера који је прихватио захтев.
- *Време одлагања реализације*. Време које прође од прихватања захтева за реализацију па до почетка реализације. програмер који је прихватио захтев за реализацију најчешће сагледава своје текуће послове и приоритете корисника који је поднео захтев и самог захтева.
- *Време реализације*. Време које се односи на посао реализације послова обухваћених захтевом корисника. Време реализације почиње од *датума почетка реализације захтева* и траје до *датума завршетка реализације захтева*. Ово време укључује радне сате које програмер проведе на реализацији захтева (у предузећу, на интернету и код корисника).
- *Време процесирања захтева*. Време од пријема захтева до завршетка свих техничких послова везаних за захтев. Време процесирања не укључује административне послове који се односе на радни налог придружен захтеву.



Слика 2. Временско процесирање захтева корисника

Животни циклус захтева корисника је време од пријема захтева па све док се заврше сви послови везани за захтев, и технички и адмистративни. Крај животног циклуса захтева је везан за затварање радног налога додељеног захтеву. Овде је важно приметити да један радни налог може обухватити више захтева корисника, што је уобичајена пракса због наплаћивања услуга кориснику.

У пракси *време реализације* захтева није једнако збиру радних сати које је програмер евидентирао да је утрошио на реализацију захтева. Разлог за то је ангажовање програмера на више послова истовремено, тако да он време унапред планира за сваки од послова који су му додељени. Формула 1 представља број стварно утрошених радних сати на реализацији захтева.

$$TWH = CWH + CSWH + IWH \quad (1)$$

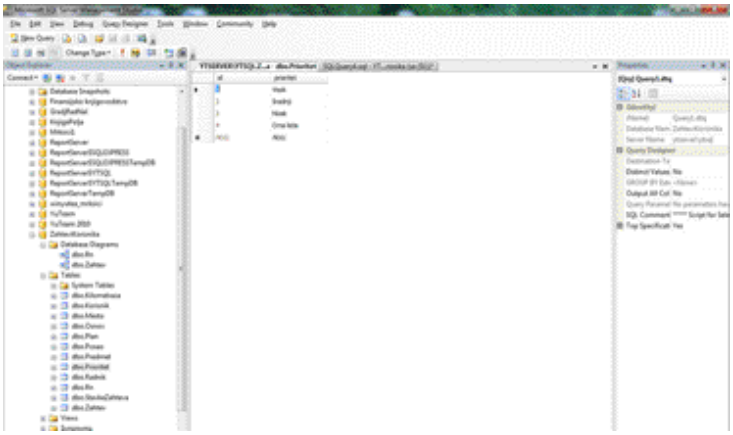
У формули 1 су уведене следеће ознаке: *Total Working Hours* (*TWH*) - укупан број радних сати утрошених за реализацију захтева; *Company Working Hours* (*CWH*) - број радних сати које је програмер провео радећи у предузећу; *Client Side Working Hours* (*CSWH*) - број радних сати које је програмер провео радећи код корисника, и *Internet Working Hours* (*IWH*) - број радних сати које је програмер провео радећи преко интернета.

Реализација и примена техничког решења

Као подршка активностима у предузећу је развијена интерна веб базирана софтверска апликација која користи репозиторијум приказан у претходној секцији. Софтверска апликација је доступна само запосленима у предузећу, и то за сада само када су у предузећу.

Репозиторијум је реализован као релациона база података постављена на *Microsoft SQL Server 2008 Express* серверу који је доступан као бесплатна верзија сервера (погледати слику 3). Овакво решење за сервер је одабрано из два разлога:

- Сервер је бесплатан и погодан за примену у веб апликацијама, а такође задовољава неопходне услове заштите података. Од ове верзије сервер обезбеђује подршку за полу-структуриране податке, укључујући и формате за дигиталне слике, аудио, видео и остале мултимедијалне формате.
- Предузеће планира да у будућности изврши миграцију својих пословних софтверских решења на *Microsoft .NET* платформу која се ослања на овај сервер, па је развој овакве интерне апликације искоришћен као упознавање са платформом пре реализације миграције пословних решења.



Слика 3. Репозиторијум захтева имплементиран на Microsoft SQL Server 2008

Веб апликација за рад са захтевима корисника је развијена применом *ASP.NET framework*-а који се користи за развој динамичких веб сајтова, веб апликација и веб сервиса на *Microsoft* платформи. Овај радни оквир омогућује израду веб страница, које се називају веб форме (*Web Forms*), које су сачињене од контрола које су сличне контролама у стандардном корисничком интерфејсу на *Windows* платформи. Ове веб контроле функционишу слично као и контроле на стандардној *Windows* платформи, што олакшава развој или миграцију на ову платформу за програмере са искуством на стандардној *Windows* платформи.

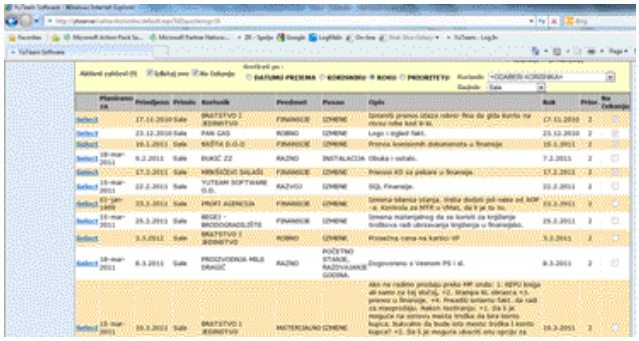
Веб апликација је реализована као вишеслојна клијент-сервер апликација, и омогућује ауторизацију сваког корисника приликом пријављивања. Веб форме садрже мени који омогућује навигацију кроз апликацију и приступ свим функционалностима које нуди апликација. Основни кориснички интерфејс се састоји од следећих веб форми:

- *Основна страна* (слика 4). Основна страница садржи преглед свих активних захтева тренутно улогованог корисника.
- *Страна за унос захтева* (слика 5). Страница на којој тренутно пријављени радник уноси нови захтев. При томе се *датум преузимања захтева* (Zahtev::datumPreuzimanja) аутоматски поставља. На овој страници се поред уноса података за захтев може извршити и прослеђивање захтева. Страница се отвара такође приликом прегледа захтева и омогућује затварање захтева када се комплетирају послови везани за захтев, а тада програмер поставља *датум завршетка реализације захтева*} (Zahtev::datumZavrsetka).
- *Страна за преглед захтева* (слика 6). Ова страна омогућује да се прегледају сви актуелни захтеви у предузећу. Захтев ће припадати одређеној групи, на пример *Нераспоређени захтеви*, *Неприхваћени захтеви* или *Недовршени захтеви* у зависности од фазе у којој се тренутно налази. Припадност одређеној групи захтева значи да су одређени датуми у току процесирања захтева постављени, а одређени нису постављени. На пример, ако се захтев налази у групи *Недовршени захтеви* то значи да је неко од програмера прихватио захтев, али да га није реализовао, што значи да је *датум прихватања захтева* (Zahtev::datumPrihvatanja) постављен, а да *датум завршетка реализације захтева* (Zahtev::datumZavrsetka) није постављен.
- *Страна за унос радног налога*. Страна омогућује креирање и рад на радном налогу који се придружује захтеву који је тренутно активан. Програмер не може креирати радни налог ако пре тога није

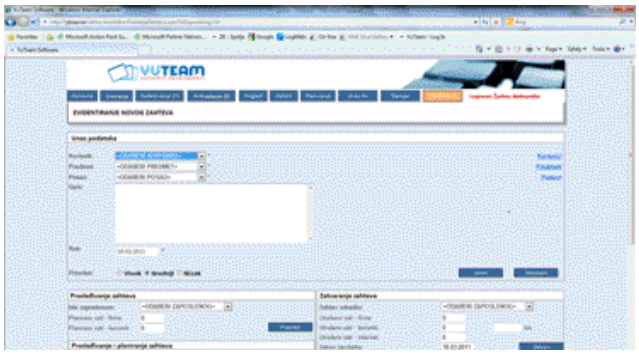
комплетирао све послове везане за захтев корисника. То значи да су у том тренутку сви датуми постављени, укључујући и *датум завршетка реализације захтева* (Zahtev::datumZavrsetka). Када програмер сними радни налог поставља се и *датум радног налога* (RadniNalog::datum), а то значи да је завршен животни циклус захтева.

Предвиђени рок за реализацију захтева (Zahtev::datumRoka) се може поставити приликом евидентирања захтева. То је рок који постави корисник. Ипак, у највећем броју случајева тај рок се не наводи. Због тога је датум *рока за реализацију захтева* на слици 2 приказан испрекиданом линијом, и разликује се од осталих датума у животном циклусу. Пограмер који прихвати захтев за реализацију ће захтев решити према дефинисаним приоритетима захтева и корисника, а ако је рок постављен тада ће са корисником усагласити детаље реализације.

Прецизна и систематска евиденција захтева, са јасно одређеним датумима у животном циклусу захтева омогућује да се изврши детаљна статистичка анализа (анализа трендова или регресија (Buglear, 2001)) процеса управљања захтевима корисника и да се на основу добијених резултата анализе уоче ндеостаци у процесу и правци будућих унапређења процеса.



Слика 4. Основна страница веб апликације



Слика 5. Страна за унос новог захтева

Слика 6. Страница за преглед захтева

ЛИТЕРАТУРА

Alain Abran, Pierre Bourque, Robert Dupuis, James W. Moore, and Leonard L. Tripp, editors. *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway, NJ, USA, 2004 version edition, 2004.

A. April, J. Bouman, A. Abran, and D. Al-Shurougi. *Software maintenance in a service level agreement: Controlling the customers expectations*. In FESMA-AEMES Software Measurement Conference 2000, Madrid, Spain, October 18-20 2000.

Alain April. *Studying supply and demand of software maintenance and evolution services*. In Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology , 352-357, 2010. DOI: 10.1109/QUATIC.2010.65.

Alain April, Jane Hu_man Hayes, Alain Abran, and Reiner Dumke. *Software maintenance maturity model (smmm): the software maintenance process model*. Journal of Software Maintenance and Evolution: Research and Practice, 17(3):197-223, 2005. DOI: 10.1002/smr.311.

Pankaj Bhatt, Gautam Shroff, and Arun K. Misra. *Dynamics of software maintenance*. ACM SIGSOFT Software Engineering Notes, 29(5):1-5, 2004. DOI: 10.1145/1022494.1022513.

Barry Boehm and Victor R. Basili. *Software defect reduction top 10 list*. Computer, 34(1):135-137, January 2001. DOI: 10.1109/2.962984.

Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language user guide*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1999.

Jacques Bouman, Jos Trienekens, and Mark Van der Zwan. Speci_cation of service level agreements, clarifying concepts on the basis of practical research. In Proceedings of the Software Technology and Engineering Practice, 169-178, 1999. DOI: 10.1109/STEP.1999.798790.

Lionel C. Briand, Victor R. Basili, Yong-Mi Kim, and Donald R. Squier. *A change analysis process to characterize software maintenance projects*. In Hausi A. Muller and Mari Georges, editors, Proceedings of the International Conference on Software Maintenance, ICSM '94, 38-49, 1994.

John Buglear. *Stats means business: a guide to business statistics*. Buttworth-Heinemann, Oxford, UK, 2001.

Gerry Coleman and Rory O'Connor. *Investigating software process in practice: A grounded theory perspective*. Journal of Systems and Software, 81(5):772-784, 2008. DOI: 10.1016/j.jss.2007.07.027.

Tore Dyba, Torgeir Dingsoyr, and Nils Brede Moe. *Process Improvement in Practice - A Handbook for IT Companies*, volume 9 of International Series in Software Engineering. Kluwer Academic Publishers, Norwell, MA, USA, 2004. DOI: 10.1007/b116193.

Capers Jones. *Software Engineering Best Practices*. McGraw-Hill, Inc., New York, NY, USA, 2010.

Magne Jorgensen. *Experience with the accuracy of software maintenance task effort prediction models*. IEEE Transactions on Software Engineering, 21(8):674-681, 1995. DOI: 10.1109/32.403791.

Gladston Aparecido Junio, Marcelo Nassau Malta, Humberto de Almeida Mossri, Humberto T. Marques-Neto, and Marco Tulio Valente. *On the benefits of planning and grouping software maintenance requests*. In 15th European Conference on Software Maintenance and Reengineering, 55-64, 2011. DOI: 10.1109/CSMR.2011.10.

Huzefa Kagdi, Michael L. Collard, and Jonathan I. Maletic. *A survey and taxonomy of approaches for mining software repositories in the context of software evolution*. Journal of Software Maintenance and Evolution: Research and Practice, 19(2):77-131, March 2007. DOI: 10.1002/smr.344.

Huzefa Kagdi, Malcom Gethers, Denys Poshyvanyk, and Maen Hammad. *Assigning change requests to software developers*. Journal of Software: Evolution and Process, 24(1):3-33, 2011. DOI: 10.1002/smr.530.

Mira Kajko-Mattsson. *Maintenance at ABB (II): Change execution processes*. In ICSM '99: Proceedings of the IEEE International Conference on Software Maintenance, 307-315, 1999. DOI: 10.1109/ICSM.1999.792628.

Mira Kajko-Mattsson, Ulf Westblom, Stefan Forssander, Gunnar Andersson, Mats Medin, Sari Ebarasi, Tord Fahlgren, Sven-Erik Johansson, Stefan Tornquist, and Margareta Holmgren. *Taxonomy of problem management activities*. In CSMR '01: Proceedings of the Fifth European Conference on Software Maintenance and Reengineering, 1-10, Lisbon, Portugal, March 14-16 2001. DOI: 10.1109/.2001.914962.

Barbara A. Kitchenham, Guilherme H. Travassos, Anneliese von Mayrhauser, Frank Niessink, Norman F. Schneidewind, Janice Singer, Shingo Takada, Risto Vehvilainen, and Hongji Yang. *Towards an ontology of software maintenance*. Journal of Software Maintenance: Research and Practice, 11(6):365-389, 1999. DOI: 10.1002/(SICI)1096-908X(199911/12)11:6<365::AID-SMR200>3.0.CO;2-W.

Claude Y. Laporte, Simon Alexandre, and Alain Renault. *Developing international standards for very small enterprises*. Computer, 41(3):98-101, March 2008. DOI: 10.1109/MC.2008.86.

Claude Y. Laporte, Alain Renault, Simon Alexandre, and Tanin Uthayanaka. *The application of iso/iec jtc 1/sc7 software engineering standards in very small enterprises*. ISO Focus, pages 36-38, September 2006.

B. P. Lientz, E. B. Swanson, and G. E. Tompkins. *Characteristics of application software maintenance*. Communications of the ACM, 21(6):466-471, 1978. DOI: 10.1145/359511.359522.

Francisco J. Pino, Francisco Ruiz, Felix Garcia, and Mario Piattini. *A software maintenance methodology for small organizations: Agile_mantema*. Journal of Software: Evolution and Process, 24(8):851-876, 2012. DOI: 10.1002/smr.541.

Ita Richardson and Christiane Gresse von Wangenheim. *Guest editors' introduction: Why are small software organizations different?* IEEE Software, 24(1):18-22, 2007. DOI: 10.1109/MS.2007.12.

Ilmari Saastamoinen and Markku Tukiainen. *Software process improvement in small and medium sized software enterprises in eastern Finland: A state-of-the-practice study*. In Torgeir Dingsoyr, editor, Software Process Improvement, volume 3281 of Lecture Notes in Computer Science, 69-78. Springer Berlin / Heidelberg, 2004. DOI: 10.1007/978-3-540-30181-3_7.

Stephen R. Schach and Amir Tomer. *A maintenance-oriented approach to software construction*. Journal of Software Maintenance, 12(1):25-45, January 2000. DOI: 10.1002/(SICI)1096-908X(200001/02)12:1<25::AIDSMR203>3.0.CO;2-2.

Ian Sommerville. *Software engineering (6th ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

Zeljko Stojanov. *Using qualitative research to explore automation level of software change request process: A study on very small software companies*. Scientific Bulletin of The Politehnica University of Timisoara, Transactions on Automatic Control and Computer Science, 57(1):31-40, March 2012. ISSN 1224-600X.

Zeljko Stojanov, Dalibor Dobrilovic, and Vesna Jevtic. *Identifying properties of software change request process: Qualitative investigation in very small software companies*. In Proceedings of the 9th IEEE International Symposium on Intelligent Systems and Informatics, SYSI 2011, 47-52, Subotica, Serbia, 8-10 September 2011. DOI: 10.1109/SISY.2011.6034369.

Philip S. Taylor, Des Greer, Gerry Coleman, Kevin McDaid, and Frank Keenan. *Preparing small software companies for tailored agile method adoption: Minimally intrusive risk assessment*. Software Process: Improvement and Practice, 13(5):421-437, 2008. DOI: 10.1002/spip.358.

Christer Thorn and Thomas Gustafsson. *Uptake of modeling practices in SMEs: initial results from an industrial survey*. In Proceedings of the 2008 international workshop on Models in software engineering , MiSE '08, 21-26, 2008. DOI: 10.1145/1370731.1370737.

Christiane Gresse von Wangenheim, Alessandra Anacleto, and Clenio F. Salviano. *Helping small companies assess software processes*. IEEE Software, 23(1):91-98, January 2006. DOI: 10.1109/MS.2006.13.

Liguo Yu. *Indirectly predicting the maintenance effort of open-source software*. Journal of Software Maintenance and Evolution: Research and Practice, 18(5):311-332, 2006. DOI: 10.1002/smr.335.

Република Србија – АП Војводина
Универзитет у Новом Саду
Технички факултет «Михајло Пупин»
Зрењанин, Ђуре Ђаковића бб
www.tfzr.uns.ac.rs
Тел.023/550-515 факс: 023/550-520
ПИБ: 101161200

Дел.бр: 04 – 3926/5
Датум: 12.12.2012.

ИЗВОД ИЗ ЗАПИСНИКА

са 60. седнице Наставно – научног већа Техничког факултета "Михајло Пупин" у Зрењанину, одржане 12.12.2012. године.

Непотребно изостављено!

5.
ИЗВЕШТАЈИ КАТЕДРИ

ПРЕДЛОГ РЕЦЕНЗЕНАТА ЗА ТЕХНИЧКО РЕШЕЊЕ

5.5.
Након уводне речи проф. др Милана Павловића, председника већа и разматрања предлога **Катедре за Информационе технологије**, гласањем, једногласно је донета

О Д Л У К А

Наставно – научно веће Техничког факултета «Михајло Пупин» у Зрењанину прихвата да се за верификацију техничког решења „**Софтверска апликација за руковање захтевима корисника која омогућује унапређење послова одржавања софтвера у малом софтверском предузећу**" именују следећи рецензенти:

- **Бранко Милосављевић**, ванредни професор, Факултет техничких наука, Нови Сад, Универзитет у Новом Саду. Научна област: Примењене рачунарске науке и информатика.
- **Зоран Шеварац**, доцент, Факултет организационих наука, Београд, Универзитет у Београду. Научна област: Софтверско инжењерство.

Аутори техничког решења су:

- **Миша Варга**, YuTeam Software ОД Зрењанин
- **Горан Николић**, YuTeam Software ОД Зрењанин
- **Жељко Стојанов**, Технички факултет "Михајло Пупин" Зрењанин, Универзитет у Новом Саду
- **Александар Жарков**, YuTeam Software ОД Зрењанин

Техничко решење је развијено у оквиру пројекта **Развој софтверских алата за анализу и побољшање пословних процеса**, број пројекта **ТР–32044**, који је финансиран од стране Министарства просвете и науке Републике Србије.

За тачност извода оверио
Драгана Бугарчић

ПРЕДСЕДНИК
НАСТАВНО – НАУЧНОГ ВЕЋА

Проф. др Милан Павловић

Доставити:
1. Рецензентима
2. Ауторима
3. Министарству просвете и науке Републике Србије, Београд
4. Архиви

Мишљење рецензената о техничком решењу

Софтверска апликација за руковање захтевима корисника која омогућује унапређење послова одржавања софтвера у малом софтверском предузећу

У оквиру предузећа **YuTeam Software OD Зрењанин** развијена је софтверска апликација за евидентирање корисничких захтева у оквиру одржавања софтвера и других послова који се односе на корисничке захтеве. Софверска апликација има веб-базирани кориснички интерфејс, а користи и репозиторијум који је реализован помоћу релационе базе података.

Пројекат технолошког развоја **ТР–32044 "Развој софтверских алата за анализу и побољшање пословних процеса"** на Техничком факултету „Михајло Пупин" у Зрењанину је обухватио, између осталог, и анализу праксе процесирања захтева. Анализа је обухватила евидентиране захтеве у периоду од маја 2010. године до новембра 2011. године (укупно 19 месеци), што је укупно 1901 захтев корисника за одржавањем. Анализом захтева је уочено да је неопходно унапредити софтверску апликацију тако да се омогући ефикаснијо процесирање захтева у зависности од статуса (приоритета) корисника, као и да се обезбеди праћење временских фаза у процесирању захтева, као и утрошка радних сати програмера (оптерећење програмера).

На основу наведене анализе извршена је модификација репозиторијума и софтверске апликације тако да се обезбеде следеће функције:

- праћење временских фаза у процесирању захтева,
- праћење оптерећења програмера и
- класификација (приоритизација) захтева на основу дефинисаног статуса корисника који пријављује захтев.

Техничко решење је верификовано у реалном радном окружењу предузећа YuTeam Software OD Зрењанин. Техничко решење се користи у свакодневном раду у предузећу за управљање захтевима корисника.

Према Правилнику о поступку и начину вредновања, и квантитативном исказивању научно-истраживачких резултата истраживача, који је донео Национални савет за научни и технолошки развој Републике Србије 2008. године ("Сл. гласник РС", бр. 38/2008) предлажемо да се приказано техничко решење прихвати као решење у категорији **М85 Софтвер (уз доказ)**.

Децембар 2012. године

Проф. Др Бранко Милосављевић

Универзитет у Новом Саду, Факултет техничких наука

Доц. Др Зоран Шеварац,

Универзитет у Београду, Факултет организационих наука

YuTeam Software OD

Pupinova 16, I sprat
23000 Zrenjanin
Tel/fax: **023/510-986, 236-122**
Tekući račun: 160-114774-07
E-mail: **office@yuteam.co.rs**
www.YuTeam.co.rs

Предмет:

Оцена техничког решења

Софтверска апликација за руковање захтевима корисника која омогућује унапређење послова одржавања софтвера у малом софтверском предузећу

које је реализовано у оквиру пројекта "*Развој софтверских алата за анализу и побољшање пословних процеса*" (бр. пројекта ТР –32044).

У току 2011. и 2012. године запослени фирме YuTeam Software OD, из Зрењанина, су активно учествовали са члановима пројектног тима у реализацији пројекта. Циљ ове сарадње је анализа и унапређење процеса управљања захтевима корисника. Један од резултат овог рада је техничко решење које се односи на модификацију софтверске апликације за управљање захтевима корисника. Ова модификација обезбеђује приоритизацију захтева на основу приоритета (статуса) корисника и детаљно праћење процеса захтева током његовог животног циклуса.

С обзиром да су у изради овог решења активно учествовали и запослени предузећа YuTeam Software ОД Зрењанин, као и чланови пројектног тима са Техничког факултета "Михајло Пупин" из Зрењанина, ово решење у потпуности је усклађено са техничким и организационим потребама предузећа. YuTeam Software ОД Зрењанин је задовољан нивоом успостављене сарадње и резултатима који су постигнути, а ово решење је један од конкретних резултата те сарадње. Решење се користи у свакодневној пракси у предузећу.

У Зрењанину,
11.12.2012. године

"YuTeam Software"

Александар Жарков, Директор
YuTeam Software OD Зрењанин

Република Србија – АП Војводина
Универзитет у Новом Саду
Технички факултет «Михајло Пупин»
Зрењанин, Ђуре Ђаковића бб
www.tfzr.uns.ac.rs
Тел.023/550-515 факс: 023/550-520
ПИБ: 101161200

Дел.бр: 04 – 4123/5
Датум: 26.12.2012.

ИЗВОД ИЗ ЗАПИСНИКА

са 61. седнице Наставно – научног већа Техничког факултета
"Михајло Пупин" у Зрењанину, одржане 26.12.2012. године.

Непотребно изостављено!

5.
ИЗВЕШТАЈИ КАТЕДРИ
ПРЕДЛОГ РЕЦЕНЗЕНАТА ЗА ТЕХНИЧКА РЕШЕЊА и ПРИХВАТАЊЕ РЕЦЕНЗИЈА ТЕХНИЧКОГ РЕШЕЊА

5.12.
На основу достављене рецензије проф. др Бранка Милосављевића (Универзитет у Новом Саду, Факултет техничких наука) и доц. др Зорана Шеварца (Универзитет у Београду, Факултет организационих наука), као и достављеног мишљења корисника решења – предузећа YUTeam Software OD из Зрењанина, а према Правилнику о поступку и начину вредновања, и квантитативном исказивању научно-истраживачких резултата истраживача, који је донео Национални савет за научни и технолошки развој Републике Србије 2008. године („Сл. гласник РС", бр. 38/2008), и на основу предлога **Катедре за Информационе технологије**, гласањем, једногласно је донета

О Д Л У К А

Наставно – научно веће Техничког факултета «Михајло Пупин» у Зрењанину, прихвата да се техничко решење под називом **„Софтверска апликација за руковање захтевима корисника која омогућује унапређење послова одржавања софтвера у малом софтверском предузећу"** аутора:

- Мише Варге, YuTeam Software OD Зрењанин,
- Горана Николића, YuTeam Software OD Зрењанин,
- Жељка Стојанова, Технички факултет "Михајло Пупин" Зрењанин, Универзитет у Новом Саду,
- Александара Жаркова, YuTeam Software OD Зрењанин

прихвати као техничко решење у категорији М85 Софтвер (уз доказ).

Образложење – Техничко решење је развијено у оквиру пројекта **"Развој софтверских алата за анализу и побољшање пословних процеса"**, број пројекта **ТР–32044**, који је финансиран од стране Министарства просвете и науке Републике Србије. Техничко решење је верификовано у реалном радном окружењу предузећа YuTeam Software OD из Зрењанина.
Према мишљењу рецензената техничко решење садржи све компоненте које чине научно истраживачки рад, а поред тога представља конкретан допринос пракси у области софтверског инжењерства пошто се користи у свакодневном раду у софтверском предузећу.

За тачност извода оверио
Драгана Бугарчић

ПРЕДСЕДНИК
НАСТАВНО – НАУЧНОГ ВЕЋА
Проф. др Милан Павловић

Доставити:
1. Рецензентима
2. Ауторима
3. Министарству просвете и науке Републике Србије, Београд
4. Архиви