Programinės įrangos priežiūrumo ir priežiūros matavimas

Sigitas Dapkūnas
Vilniaus universiteto
Matematikos ir informatikos fakulteto
Programų sistemų katedros docentas daktaras
Vilnius University,
Faculty of Mathematics and Informatics,
Department of Software Engineering,
Assoc. Professor, PhD
Naugarduko g. 24, LT-03225 Vilnius
Tel. (8 5) 219 3064
Faksas (8 5) 215 1585
El. paštas: *Sigitas.Dapkunas@mif.vu.lt*
http://www.mif.vu.lt/katedros/se/Staff/Dapkunas.htm

Valdemaras Zaramba
Vilniaus universiteto
Matematikos ir informatikos fakulteto
Informatikos katedros
magistrantas
Vilnius University,
Faculty of Mathematics and Informatics,
Department of Informatics,
postgraduate
Naugarduko g. 24, LT-03225 Vilnius
Tel. 8 622 88101
El. paštas: *valdas.zaramba@gmail.com*

Programinės įrangos priežiūra suprantama kaip programinės įrangos modifikavimas atidavus naudotojui. Programos modifikuojamos taisant rastas klaidas, gerinant įvairius programinės įrangos parametrus, plečiant arba keičiant funkcionalumą. Programinės įrangos priežiūra yra viena iš programinės įrangos gyvavimo ciklo dalių. Ilgą laiką programinės įrangos priežiūrai buvo skiriamas mažesnis dėmesys nei jos kūrimui. Pastaruoju metu padėtis pasikeitė. Vertinant ir gerinant programinės įrangos kokybę bei priežiūrą, matuojamas tiek programinės įrangos procesas, tiek programinis produktas. Straipsnyje analizuojami programinės įrangos priežiūrumo, priežiūros proceso matai, taip pat šiam tikslui skirtos programinės priemonės.

Įvadas

Baigus kurti programinį produktą ir pateikus jį į rinką, prasideda programinės įrangos gyvavimo ciklo priežiūros (palaikymo, eksploatacijos) fazė. Šioje fazėje taisomos eksploatuojant pastebėtos klaidos, gerinami įvairūs programų sistemos vykdymo parametrai, keičiamos programos pagal naujai atsiradusius užsakovo reikalavimus, plečiamas programų funkcionalumas. Pavyzdžiui, finansų valdymo ir apskaitos informacinės sistemos turi atitikti galiojančius Lietuvos Respublikos įstatymus ir įstatymų įgyvendinamąsias normas. Keičiantis įstatymams, turi būti keičiama ir informacinė sistema. Praktika rodo, kad tokie pakeitimai yra gana dažni.

Programinės įrangos priežiūra yra svarbi programinės įrangos gyvavimo ciklo fazė, nors istoriškai jai skiriama mažiau dėmesio nei kitoms fazėms (SWEBOK, 2005). Programinio produkto palaikymo svarbumą liudija ir tai, kad tarptautiniame standarte ISO/IEC 9126, skirtame programinio produkto kokybei, yra apibrėžta programinio produkto kokybės charakteristi-

ka – priežiūrumas (palaikomumas, angl. *maintainability*). Priežiūrumas suprantamas kaip galimybė programinį produktą modifikuoti: ištaisyti pastebėtas klaidas ir neatitikimus, gerinti programinę įrangą, keisti, pasikeitus operacinei terpei arba atsiradus naujiems funkciniams ir kitiems reikalavimams. Vertinant priežiūrumą pagal šį standartą, reikia parinkti tinkamas subcharakteristikas ir matus.

Bendru pavidalu priežiūrumą galima įvertinti kaip išmatuojamų priežiūrumo atributų funkciją (Land, 2002):

$$M = f(A_1, A_2, ..., A_n),$$

čia M – priežiūrumas, $A_1, A_2, ..., A_n$ – išmatuojami atributai.

Funkcija $f(A_1, A_2, ..., A_n)$ konstruojama kiekvienam konkrečiam atvejui, atsižvelgiant į atributų svarbumą ir parenkant tinkamus atributų svorius. Kiekvienu konkrečiu atveju tai priklauso nuo daugelio aplinkybių. Yra pasiūlyta daug įvairių priežiūrumo matų (Land, 2002).

Vienas iš žinomiausių matų, apimančių daugelį aspektų, yra priežiūrumo indeksas (MI – *maintainability index*). Jis grindžiamas Halsteado matais, McCabe ciklomatiniu sudėtingumu, programos kodo eilučių skaičiumi ir komentarų dydžiu. Nurodytą matą apie 1990 metus pasiūlė JAV Idaho universiteto mokslininkai. Vėliau įvairūs autoriai daug kartų jį modifikavo tobulindami pastebėtus trūkumus (Welker, 2001). Pasiūlytu matu atsižvelgiama į:

- kintamųjų kiekį ir jų naudojimą programose,
- programos logikos sudėtingumą,
- programos dydį,
- komentarų kiekį, padedantį žmonėms suprasti programas.

Programinės įrangos gyvavimo ciklo veiklų kaina nuolat auga. Kad ši kaina sumažėtų, reikalinga strategija. Šios strategijos dalimi turi būti programinės įrangos matavimas (SWEBOK, 2005). Matavimas yra būtinas procesams gerinti, o procesai turi būti tokie, kad juos būtų galima išmatuoti. Programinės įrangos priežiūros įmonėms svarbu matuoti priežiūrą. Tam naudojami įvairūs matai. Visų pirma yra matuojama dydis, pastangos, laikas ir kokybė (SWEBOK, 2005).

Programinei įrangai matuoti, matavimo rezultatams saugoti ir apdoroti dažnai naudojami programiniai įrankiai, kurie palengvina šiuos procesus. Sukaupti duomenys plačiai naudojami ne tik išvadoms apie atliktus matavimus. Taip kaupiama patirtis, kuri padeda geriau pasirengti ir atlikti naujus matavimus.

Straipsnyje nagrinėjama, kokie programinės ir įrangos priežiūrumo ir priežiūros proceso matai gali būti naudingi įmonei, kuriančiai ir palaikančiai informacines sistemas, kurias reikia dažnai keisti, kokie programiniai įrankiai gali būti tam naudojami.

Programinės įrangos priežiūrumo indeksas

Programinės įrangos priežiūrumas yra svarbi programinės įrangos kokybės charakteristika, viena iš šešių, įtrauktų į standarto ISO/IEC 9126 apibrėžtą programinio produkto kokybės modelį. Ši charakteristika atspindi sukurtos programinės įrangos galimybes būti modifikuojamai.

Priežiūrumo matavimas svarbus ne tik sukurtos programinės įrangos kokybei vertinti, bet ir programinės įrangos priežiūrai prognozuoti. Kartais, modifikuojant jau naudojamą programinę įrangą, programos tampa gana painios. Tai sunkina tolesnį jų modifikavimą ir naudojimą. Gali ateiti

toks laikas, kai programas bus pigiau perrašyti, o ne modifikuoti senas. Priežiūrumo matavimas gali padėti nustatyti ir numatyti tokios situacijos galimybę. Sistemingas priežiūrumo matavimas naudingas ir kuriant programas. Gavus blogus matavimo rezultatus, į tai galima atsižvelgti, pagerinti programos priežiūrumą ir kartu sukurti kokybiškesnį programinį produktą.

Priežiūrumo matų pasiūlyta gana daug. Jau minėtas priežiūrumo indeksas agreguoja daugelį žinomiausių programinės įrangos matų. Šis matas sukėlė didelį mokslininkų, gamybininkų, administratorių susidomėjimą. Jis remiasi programinių modulių matavimų vidurkiais (Liso, 2001). Priežiūrumo indeksui skaičiuoti naudojamas regresinis modelis. Jis sukurtas remiantis Hewletto–Packardo vykdytų programų sistemų priežiūros daugelio matavimų duomenimis.

Bene labiausiai paplitusi ši priežiūrumo indekso skaičiavimo formulė:

$$MI = 171 - 5{,}2 \cdot \ln(aveV) - 0{,}23 \cdot aveV(g') - $$
$$- 16{,}2 \cdot \ln(aveLOC) + 50 \cdot \sin\sqrt{2{,}46 \cdot perCM};$$

čia: *MI* – priežiūrumo indeksas,
aveV – vidutinė modulio Halsteado apimtis,
aveV(g') – vidutinis modulio ciklomatinis sudėtingumas,
aveLOC – vidutinis modulio kodo eilučių skaičius,
perCM – vidutinis modulio komentarų eilučių skaičius (procentais nuo bendro eilučių skaičiaus).

Pagal *MI* dydį programinė įranga vertinama kaip:

- prasto priežiūrumo, kai $MI < 65$,
- vidutinio priežiūrumo, kai $65 \le MI < 85$,
- gero priežiūrumo, kai $MI \ge 85$.

Nurodyta programinės įrangos priežiūrumo skaičiavimo formulė nėra vienintelė (Welker, 2001). Įvairūs autoriai ją keitė. Bene daugiausia ginčų kėlė ir kelia programos komentarų dėmuo. Pirmoje versijoje buvo imamas komentarų eilučių skaičius. Vėliau jį pakeitė į procentus nuo viso eilučių skaičiaus, nes modelis buvo labai jautrus komentarų eilučių skaičiui. Pavyzdžiui, jei mažame modulyje būtų daug komentarų, tai priežiūrumo indeksas labai padidėtų. Siekiant sumažinti tokį jautrumą buvo pereita prie procentų ir įvestas koeficientas 50. Yra pasiūlymų apskritai atsisakyti komentarų dėmens arba prieš nusprendžiant, ar šis dėmuo bus naudojamas, ištirti komentarus ir įvertinti jų kokybę. Pavyzdžiui, gana dažnai kaip komentaras yra nereikalingas programos kodas, kuris niekaip nepadeda suprasti programos.

Programinės įrangos priežiūros matai

Programinės įrangos priežiūros metu taisomos pastebėtos klaidos, keičiamas programų funkcionalumas, gerinamas programos efektyvumas. Eksploatuojant programinę įrangą, gaunama pranešimų apie defektus, klaidas ir kitas naudotojo problemas. Nors tokių pranešimų kiekis ir vidutinis jų gavimo laikas yra svarbūs matai, bet jie labiau charakterizuoja programinės įrangos kokybę, kuri buvo gauta programinės įrangos kūrimo fazėje, bet ne programinės įrangos palaikymo kokybę. Palaikymo fazėje reikia kuo greičiau ir kuo kokybiškiau pataisyti programinę įrangą, joje rastas klaidas ir netikslumus. Nors tai nepagerins rastų defektų rodiklio mato, bet gali padidinti užsakovo pasitenkinimą. Todėl yra svarbūs šie matai (Kan, 2003):

- neatliktų pataisymų indeksas,

- klaidos taisymo laikas ir reaguojamumas,
- neatliktų pataisymų procentas,
- pataisymų kokybė.

Neatliktų pataisymų indeksas skaičiuojamas laiko intervalui, pavyzdžiui, mėnesiui, ir parodo tiek gaunamų pranešimų apie klaidas, tiek ištaisytų klaidų kiekį:

$$I = \frac{N_i}{N_g} \times 100\%,$$

čia: I – neatliktų pataisymų indeksas,
N_i – per mėnesį ištaisytų klaidų skaičius,
N_g – per mėnesį užregistruotų klaidų skaičius.

Daugelyje programinės įrangos kūrimo organizacijų yra nustatytas laikas, per kurį reikia ištaisyti defektus. Tai paprastai priklauso nuo problemos sunkumo. Kritinėse situacijose, kai esama rizikos dėl programinės įrangos defektų, dirbama nežiūrint valandų. Kitais atvejais juos galima taisyti laisviau. Reaguojamumo matas paprastai skaičiuojamas kaip visų problemų laiko vidurkis nuo jų atsiradimo iki pataisymo. Jei yra ekstremalių reikšmių, vietoj vidurkio reikia naudoti medianą. Tokie atvejai gali pasitaikyti mažiau sudėtingose programose, kai klientas nereikalauja pataisymų ir problema gali būti ilgai nepataisyta.

Dažnai trumpas pataisymo laikas padidina kliento pasitenkinimą. Bet esama skirtumo tarp trumpo pataisymo laiko ir reaguojamumo. Vidurkio skaičiavimas gali klientui paslėpti individualius skirtumus. Reaguojamumui yra svarbūs kliento lūkesčiai, suderintas pataisymo laikas ir pasižadėjimų klientui laikymasis.

Neatliktų pataisymų procentas skaičiuojamas šia formule:

$$N_n = \frac{N_v}{N_g} \times 100\%,$$

čia: N_n – neatliktų pataisymų procentas,
N_v – skaičius pataisymų, viršijusių atlikimo laiką,
N_g – skaičius pranešimų apie klaidas, gautų per tam tikrą laiką.

Paprastai neatliktų pataisymų procentas skaičiuojamas savaitei.

Pataisymų kokybė arba defektinių pataisymų skaičius yra kitas svarbus programinės įrangos priežiūros matas. Naudotojo požiūriu blogai, jei randama funkcinių defektų. Dar blogiau, jei pataisymas yra defektinis. Pataisymas yra defektinis, jei nebuvo pašalinta rasta problema arba rastas defektas buvo pataisytas, bet dėl jo taisymo atsirado naujų defektų. Tokie defetiniai pataisymai veikia naudotojo pasitenkinimą. Defektinių pataisymų procento matas yra defektinių pataisymų procentas iš visų pataisymų per tam tikrą laikotarpį, pavyzdžiui, per vieną mėnesį.

Defektiniai pataisymai gali būti registruojami dviem būdais: tą mėnesį, kai jie rasti, arba tada, kai buvo pataisyti. Pirmu atveju tai yra naudotojo matas, antru – proceso matas. Skirtumas tarp šių dviejų datų yra sugaištas defektinio pataisymo laikas. Prasminga stebėti trukmės duomenis ir kitą informaciją, tokią kaip skaičius naudotojų, kuriuos paveikė defektinis pataisymas. Paprastai kuo ilgesnis trukmės periodas, tuo daugiau paveikia naudotojų, nes yra daugiau laiko panaudoti defektinį pataisymą savo sistemoje.

Yra argumentų prieš defektinių pataisymų procento naudojimą. Jei defektų ir kartu pataisymų yra daug, tai mažas procentas rodys optimistinį vaizdą, bet defektinių pataisymų skaičius gali būti didelis.

Todėl šiam matui reikia tiksliai suskaičiuoti defektinius pataisymus. Savaime suprantama, kad kokybės požiūriu palaikymo proceso tikslas yra nulis defektinių pataisymų.

Padarytų pataisymų dydis paprastai yra matuojamas programos kodo eilučių skaičiumi. Norint įvertinti šį dydį, reikia žinoti ne tik pridėtų, bet ir pašalintų bei modifikuotų eilučių skaičių. Todėl autonominė kodo eilučių skaičiavimo priemonė šiuo atveju nenaudinga. Ji turi būti integruota į programavimo įrankį.

Priežiūrumo indekso matavimo priemonės

Kuriant arba modifikuojant programinę įrangą patariama nuolat matuoti priežiūrumo indeksą ir atsižvelgti, į kurį intervalą – prasto, vidutinio ar gero – jis patenka. Jei priežiūrumo indeksas prastas, tai reikia spręsti, ar ne geriau tokį modulį perprogramuoti, nes tai bus pigiau, negu jį toliau modifikuoti.

Priežiūrumo indeksas yra vienas iš platesnį programinę įrangą kuriančių firmų pripažinimą įgavusių programinės įrangos priežiūrumo matų. Kai kurios programinės įrangos kūrimo firmos (pavyzdžiui, „Oracle“, „FreeBSD“, „Microsoft“, JAV Gynybos departamentas ir kt.) šį matą pripažino kaip vieną iš standartinių matų priežiūrumui matuoti (Sarwar, 2008). Todėl buvo sukurtos priemonės, automatiškai matuojančios šiuo matu. Matavimo priemonės realizuojamos įvairiai – kaip autonomiškai dirbančios programos, kaip programų kūrimo priemonės papildinys (pavyzdžiui, „JHawk“) arba iš karto integruotos į programų kūrimo priemonę (pavyzdžiui, „MS Visual Studio 2008“ ir tolesnės versijos).

Yra tiek atviro kodo, tiek uždaro kodo priežiūrumo indekso matavimo priemonių. Tokias priemones lygino Pakistano Lahore inžinerijos ir technologijos universiteto mokslininkai (Sarwar, 2008). Jie tyrė tris atviro ir keturias uždaro kodo priemones. Buvo sugalvoti palyginimo kriterijai, tirta, kaip priežiūrumo indeksą veikia įvairūs parametrai – kodo eilučių ar kodo sakinių skaičiavimas, komentarų skaičiavimas ar neskaičiavimas, programos kodo suprantamumas, ciklomatinis sudėtingumas, Halsteado dydžio arba Halsteado pastangų naudojimas formulėje.

Šių autorių požiūriu, dabar naudojamos priežiūrumo indekso matavimo priemonės gana skirtingai skaičiuoja įvairius indeksą veikiančius parametrus. Pavyzdžiui, kokiam nors parametrui skiriama daug dėmesio, o kiti ignoruojami. Todėl labai svarbu standartizuoti priežiūrumo indekso skaičiavimą. Mūsų požiūriu, tai yra sunkiai pasiekiamas tikslas. Šis matas atsirado maždaug prieš dvidešimt metų, bet iki šiol vis dar tobulinamas. Kita autorių išvada ta, kad priežiūrumas labai susijęs su programos kodo skaitomumu, bet skaičiuojant indeksą į šį parametrą neatsižvelgiama. Su šia išvada reikia sutikti.

Kitiems autoriams, dirbantiems Nyderlanduose ir keletą metų praktiškai naudojusiems priežiūrumo indeksą, atrodo, kad šis matas sukelia per daug problemų ir neduoda norimo efekto (Heitlager, 2007). Jų nuomone, svarbiausias trūkumas, kad turint priežiūrumo indekso reikšmę neaišku, kas šiai reikšmei daro didžiausią įtaką. Taigi nežinoma, ką reikia keisti, kad priežiūrumas pagerėtų.

Šio straipsnio autoriai nagrinėjo matavimo priemonę, integruotą į „Visual Studio 2010 Premium Edition“. Matavimo

priemonė matuoja pagal penkis matus. Vienas iš jų yra priežiūrumo indeksas. Dar du – ciklomatinis sudėtingumas ir kodo eilučių skaičius – tai dydžiai, kurie įeina į priežiūrumo indeksą. Likę matai – klasės paveldimumo lygmuo ir klasių priklausomybė.

Skaičiuojant indeksą, į komentarų eilutes neatsižvelgiama. Pagal išmatuotą indekso reikšmę nustatomas priežiūrumo lygis (prastas, vidutinis, geras) ir ataskaitoje pažymimas raudona, geltona arba žalia spalva. Visų matų reikšmės matomos toje pačioje ataskaitos eilutėje. Taigi matomos ne tik priežiūrumo indekso, bet ir jo dedamųjų reikšmės. Tai, mūsų manymu, padeda spręsti Nyderlandų mokslininkų (Heitlager, 2007) nurodytą problemą.

Nagrinėta matavimo priemonė gera tuo, kad matavimo rezultatus galima detalizuoti iki metodo lygmens. Todėl iš karto matyti, kurie metodai yra galbūt per ilgi, sudėtingi arba klasių paveldimumo lygmuo per gilus ir jis mažina bendrą sistemos priežiūrumą.

Programinės įrangos problemų stebėsenos sistemos

Naudojamos klaidų stebėsenos sistemos, leidžiančios rinkti informaciją apie klaidas ir stebėti, kaip į klaidas reaguojama. Klaidų stebėsenos sistemos yra atskira programinės įrangos problemų stebėsenos sistemų rūšis. Tokios sistemos paprastai naudojamos ne vien klaidoms, bet ir naujiems funkcionalumo reikalavimams, užsakovo (programinės įrangos naudotojo) problemoms fiksuoti ir stebėti. Šios sistemos padeda užtikrinti programinės įrangos kokybę. Naudojant stebėsenos sistemą galima:

- daryti įrašus – registruoti naudotojų klausimus, klaidas, naujus funkcionalumo reikalavimus, kitas naudotojo problemas, kylančias dėl nepakankamo aiškumo naudojant programą, nesupratimo ar pan.; fiksuojant klaidas galima nurodyti jų sunkumą, kas pranešė ir kitas detales;
- keisti įrašų būsenas – naujas, atidarytas, vykdomas, laukia testavimo, iš naujo atidarytas, patvirtintas ir pan.;
- fiksuoti laiką – tarp būsenų pasikeitimo, planuojamą laiką ir kt.;
- priskirti užduotis konkrečiam vykdytojui arba vykdytojų grupei;
- daryti įvairias ataskaitas – tiek tekstines, tiek diagramas.

Klaidų stebėsenos sistemos skiriasi savo funkcinėmis galimybėmis, naudojimo patogumu, pateikiamomis ataskaitomis. Tačiau galimybė fiksuoti pranešimus apie klaidas, defektus, kitas naudotojo problemas, jų atsiradimo laiką ir stebėti būsenų kitimą laiko atžvilgiu leidžia šias sistemas naudoti programinės įrangos priežiūros procesui matuoti. Šios sistemos taip pat gali būti naudojamos klaidas taisančių programuotojų produktyvumui matuoti. Tik šis matavimas nėra paprastas, nes klaidos smarkiai skiriasi pagal sudėtingumą, o nuo to labai priklauso jų taisymo laikas.

Priežiūros proceso matavimui nagrinėti pasirinktos dvi sistemos: „StarTeam“ (StarTeam, 2011) ir JIRA (JIRA, 2011). Abu produktai turi panašų bazinį funkcionalumą, skiriasi problemų registravimo būdai, pateikiamos ataskaitos. Abiejose sistemose ataskaitas galima gauti diagramų arba lentelių pavidalu.

Sistemoje „StarTeam“ vartotojo problemų įrašus galima susieti tiesiogiai su projektu, katalogu, pakatalogiu, failu ar klase. Kiekvieną užduotį galima priskirti konkrečiam vykdytojui. Tokia galimybė

leidžia matuoti palaikymą visai programai ar jos dalims, atskiriems moduliams. Įrašus, esančius pakeitimų reikalavimų arba defektų sąraše, galima susieti tarpusavyje. Abiejuose įrašų tipuose galima nurodyti nuorodą į kitą įrašą. Toks „StarTeam“ funkcionalumas yra naudingas tada, kai dėl vienų klaidų pataisymo atsiranda naujos arba pridėtas naujas funkcionalumas sugriauna tinkamai veikiantį kitą. Šią galimybę galima panaudoti pataisymo kokybei matuoti. Nuorodomis galima atsekti, kokio tipo pakeitimai sukėlė naujas klaidas. Skaičiuojant tokias klaidas, galima susidaryti pakeitimų rizikos grupes ir ateityje skirti daugiau dėmesio testuojant rizikos grupės pakeitimus.

Sistemoje „StarTeam“ automatiškai skaičiuojama pagal du matus: klaidos taisymo reaguojamumą ir pavėluotų (neatliktų) pataisymų procentą. Matavimų rezultatus pagal kitus matus galima apskaičiuoti. Tam reikia generuojant ataskaitą nurodyti tinkamus filtravimo kriterijus ir suskaičiuoti pagal matus aprėžiančias formules. Pavyzdžiui, skaičiuojant neatliktų pataisymų indeksą, reikia žinoti per mėnesį ištaisytų ir per mėnesį užregistruotų klaidų skaičius. Ištaisytų klaidų skaičius – tai sistemos įrašai, kurie per mėnesį įgavo reikšmę „problema išspręsta“. Užregistruotų klaidų skaičius – tai tą mėnesį atsiradę būsenos „naujas“ įrašai. Nurodžius tinkamus filtravimo kriterijus, galima gauti šiuos skaičius.

Filtruojant yra galimybė išrinkti duomenis pagal vykdytojus. Todėl matuoti galima ne visam projektui, o kiekvienam vykdytojui. Tokie matavimai leidžia įdėmiau stebėti projekto eigą. Pamačius rezultatų blogėjimą, reikia analizuoti tokio blogėjimo priežastis. Pavyzdžiui, ar modulis, kurį taiso konkretus asmuo, yra labai sudėtingas, ar darbuotojas pradėjo prasčiau dirbti? Aišku reikia atsižvelgti ir į pataisymų kokybę, t. y. kiek „ištaisytų“ klaidų grįžta į klaidų stebėjimo žurnalą.

Konkrečią datą galima pasižiūrėti, kokia pataisymų procentinė dalis jau padaryta, ir prognozuoti, ar esant tokiam produktyvumui viskas bus padaryta laiku. Nustačius potencialų vėlavimą, galima stengtis vėlavimą sumažinti. Tyrimo metu išbandyti keli būdai gresiančiam vėlavimui sumažinti:

1. Ilginti darbo valandas. Kaip parodė tyrimas, tai yra bloga praktika, nes paskutinę akimirką daromi darbai papildomomis darbo valandomis dažniausiai vėl sugrįždavo į defektų žurnalą. Dėl neatidumo buvo daromi kiti defektai, sukuriamos naujos problemos.
2. Įvertinti, kurios užduotys yra svarbiausios (aukščiausio prioriteto), ir jas baigti iki termino pabaigos, kitus pataisymus perkelti į vėlesnę versiją. Tokiu būdu pavėluotų pataisymų procentas nesumažėja, tačiau neprarandamas klientų pasitenkinimas.
3. Iš likusių įrašų uždaryti tuos, kurie yra padaromi greičiausiai, yra nepasiteisinusi praktika. Pavėluotų pataisymų procentas būdavo sumažinamas iki minimumo, tačiau likdavo didelių klaidų, dėl kurių kartais neveikdavo atskiros sistemos funkcijos. Klientų pasitenkinimas smarkiai krisdavo, daugėdavo užsakovų skambučių ir naujų defektų skaičius.
4. Įvertinus likusias užduotis, išrinkti svarbiausias, aukščiausio prioriteto. Paskaičiavus, kiek laiko užtruks jas

atlikti, pakoreguoti versijos išleidimo datą taip, kad būtų įmanoma spėti atlikti užduotis.

Taigi iš visų bandytų metodų geriausi buvo antras ir ketvirtas. Pirma taisomos kritinės, aukščiausio prioriteto užduotys, o jei nespėjama, pakoreguojama versijos išleidimo data. Tokiu atveju klientų pasitenkinimas didžiausias.

JIRA yra projektų valdymo sistema, skirta programų sistemų kūrimo eigos stebėsenai, darbų planavimui. Klaidų stebėsena yra viena iš daugelio jos vykdomų funkcijų. Įvairius įvykius galima stebėti tiek pagal projektus, projektų dalis, tiek pagal grupes arba pogrupius, dalyvaujančius juos kuriant.

Sistema JIRA leidžia generuoti įvairias ataskaitas diagramų ir lentelių pavidalu. Jas galima eksportuoti įvairiais formatais, pavyzdžiui, į „MS Excel“, ir naudoti tolesniems skaičiavimams. Duomenims išrinkti naudojama JIRA užklausų kalba. Dauguma ataskaitų skirtos projekto eigai stebėti. Keletas iš jų susijusios su užregistruotų problemų, klaidų taisymu ir gali būti panaudotos priežiūros procesui matuoti.

Su programinės įrangos priežiūra susijusios ataskaitos arba matuoja pagal anksčiau aptartus matus, arba leidžia lengvai pagal juos apskaičiuoti. Užregistruotų ir išspręstų problemų atskaita suskaičiuoja, kiek per nurodytą laiko tarpą buvo užregistruota ir kiek išspręsta problemų. Duomenys vaizduojami diagrama ir pateikiami lentelėje. Pagal šios lentelės duomenis nesunku suskaičiuoti neatliktų pataisymų indeksą. Klaidos taisymo laiko atskaita iš karto matuoja pagal programinės įrangos palaikymo matą. Taip pat galima sužinoti nepataisytų klaidų vidutinį amžių kiekvienai pasirinkto intervalo dienai.

Išvados

Programų sistemų priežiūra yra svarbi programinės įrangos gyvavimo ciklo dalis. Ypač tai aktualu įmonėms, kurančioms programinę įrangą, kurią dažnai reikia keisti dėl to, kad keičiasi įvairūs reikalavimai, pavyzdžiui, buhalterinę veiklą, mokesčius reglamentuojantys įstatymai. Kuriama programinė įranga turi būti kuo lengviau modifikuojama. Taip pat svarbu efektyviai organizuoti programinės įrangos priežiūros procesą. Programinės įrangos matavimai padeda objektyviai įvertinti situaciją ir priimti naudingus sprendimus. Programinės įrangos inžinerijos mokslininkai pasiūlė įvairių matų programinei įrangai matuoti. Atėjo toks laikas, kai pasiūlyti matai pradėti taikyti praktikoje. Straipsnyje išnagrinėti kai kurie matai, susiję su programinės įrangos priežiūrumo ir priežiūros proceso matavimu. Analizės pagrindu galima daryti šias išvadas:

1. Matuojant programinės įrangos priežiūrumą, geriausias rezultatas gaunamas, kai matavimas yra integruotas į programinės įrangos kūrimo priemones. Tai padeda taip kurti programinę įrangą, kad vėliau ją būtų lengviau modifikuoti.
2. Klaidų stebėsenos sistemos padeda efektyviai organizuoti programinės įrangos priežiūros darbą. Šių sistemų generuojamos ataskaitos leidžia gauti reikšmes pagal mokslininkų pasiūlytus matus arba duomenis eksportuoti įvairiais formatais. Eksportavus, pavyzdžiui, į skaičiuoklės dokumentą, iš surinktų duomenų galima paprastai apskaičiuoti kitus reikalingus dydžius.

LITERATŪRA

HEITLAGER, Ilja; KUIPERS, Tobias; VISSER, Joost (2007). A Practical Model for Measuring Maintainability. In *QUATIC'07 Proceedings of the 6th International Conference on Quality of Information and Communications Technology*, p. 30–39.

JIRA issue and project tracking [žiūrėta 2011 m. balandžio 12 d.]. Prieiga per internetą: <http://www.atlassian.com/software/jira/>.

KAN, Stephen H. (2003). *Metrics and Models in Software Quality Engineering*. Addison-Wesley. 528 p. ISBN 0201729156.

LAND, Rikard (2002). Measurements of Software Maintainability. In *Proceedings of ARTES Graduate Student Conference*, p. 23–27 [žiūrėta 2011 m. balandžio 12 d.]. Prieiga per internetą: <http://www.artes.uu.se/events/gsconf02/papers/Land_Maintainability.pdf>.

LISO, Aldo (2001). Software Maintainability Metrics Model: An Improvement in the Coleman-Oman Model. *CrossTalk* August, p. 15–17.

StarTeam. Borland StarTeam [žiūrėta 2011 m. balandžio 12 d.]. Prieiga per internetą: <http://techpubs.borland.com/starteam/>.

SARWAR, Muhammad Imran; TANVEER, Wasif; SARWAR; Imran; MAHMOOD, Waqar (2008). A Comparative Study of MI Tools: Defining the Roadmap to MI Tools Standardization. In *Proceedings of the 12th IEEE International Multitopic Conference*, December 23–24, p. 379–385.

(SWEBOK, 2005). ISO/IEC TR 19759:2005 Software Engineering – Guide to the Software Engineering Body of Knowledge [žiūrėta 2011 m. balandžio 18 d.]. Prieiga per internetą: <http://www.computer.org/portal/web/swebok/htmlformat>.

WELKER, Kurt D. (2001). The Software Maintainability Index Revisited. *CrossTalk*, August, p. 18–21.

SOFTWARE MAINTAINABILITY AND MAINTENANCE MEASUREMENT

Sigitas Dapkūnas, Valdemaras Zaramba

S u m m a r y

Software maintenance means software modification after its delivery to customers. The programs are modified with the purpose to correct faults, to improve performance or other program parameters, to enhance or change its functionality. Software maintenance is an integral part of a software life cycle. Historically, software maintenance received less attention than software development in most organizations. Recently, the situation has changed. For software quality or software maintenance assessment and improvement, software process or software product measurement is used. The paper deals with software maintenance measurement. It analyses software maintainability and maintenance measures, the measurement tools.