Kokybės užtikrinimo veiklos, įrankiai ir metrikos

Rūta Bartkevičiūtė
Kokybės inžinierė
ruta.bartkeviciute@bpi.lt

Kas yra kokybė?

- Kokybė yra tinkamumas pagal paskirtį [*vartotojo požiūris*]
- Kokybė yra specifikacijos atitikimas [*gamybinis požiūris*]
- Kokybė siejama su būdingomis produkto savybėmis [*produkto požiūris*]
- Kokybė yra susijusi su pinigų suma, kurią vartotojas nori sumokėti [*vertės požiūris*]
- Kokybė yra atitikimas vartotojo lūkesčiams *[pasitenkinimo požiūris*]

Kas yra PĮ kokybė?

IEEE: Mastas, kai sistema, komponentas arba procesas atitinka: (1) nustatytus reikalavimus, bei (2) vartotojo arba užsakovo poreikius arba lūkesčius

Kokybės charakteristikos

Efektyvumas

Korektiškumas

Lankstumas

Integruojamumas

Saugumas

Pernaudojamumas

Portabilumas

Patogumas

Patikimumas

Testuojamumas

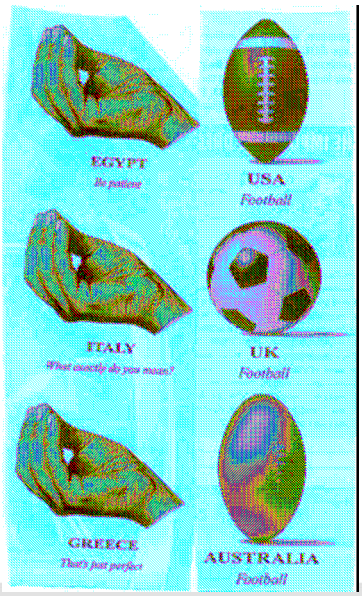
Kas yra testavimas?

- Sistemos ar programos operacijos vykdymas arba simuliavimas
- Programos analizė, siekiant atrasti klaidas
- Programos ir projekto artefaktų atributų įvertinimas ir nustatymas, ar buvo pasiekti reikalaujami ar priimtini rezultatai
- Reikalavimų ir dizaino inspekcijos, peržiūros poromis, o taip pat testavimo kodo vykdymas

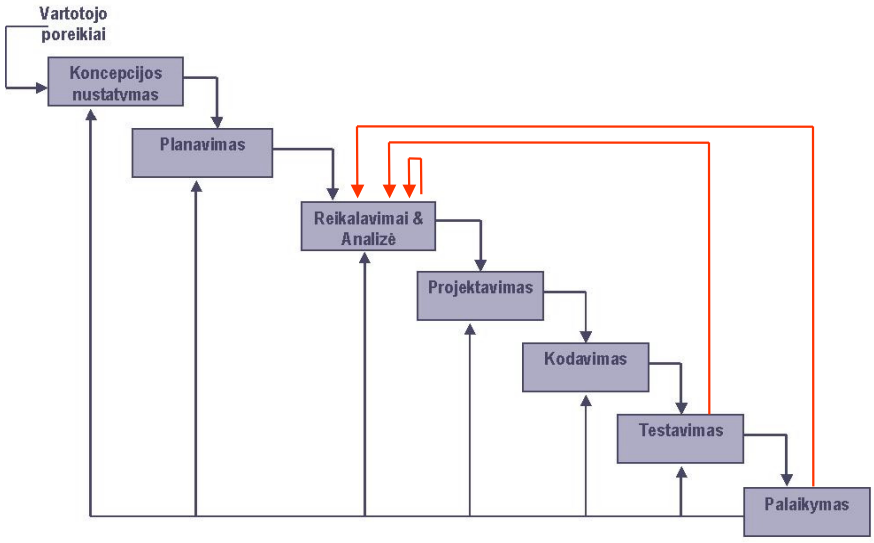
Testavimo būtinybė ir kaina

Kaina – kuo vėliau rastas defektas, tuo daugiau kainuoja jį ištaisyti
Kokybė – surastas defektas gali būti ištaisytas.
Programuotojai negali sukurti programos be defektų
Programuotojai nėra geri testuotojai

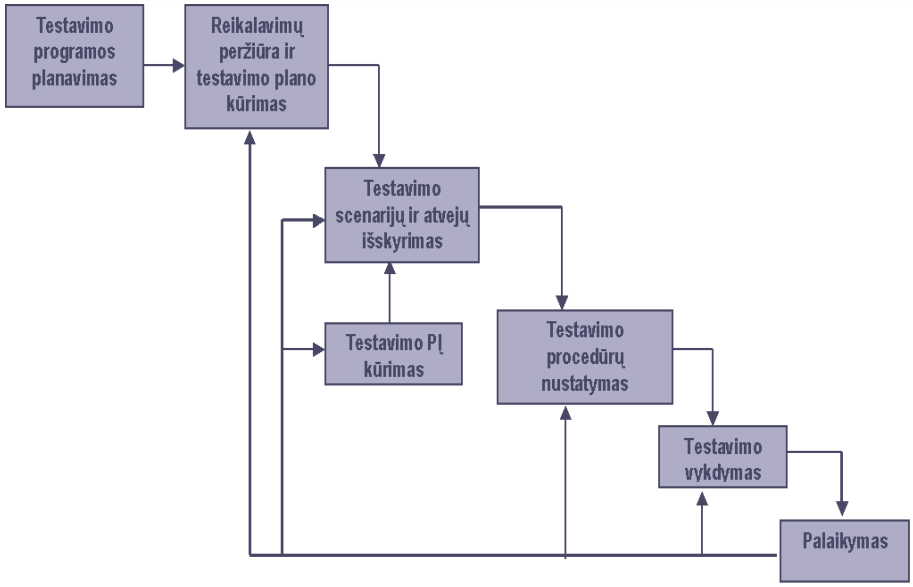
Ar vienodai suprantam apie ką kalbam?



Kodėl defekto ištaisymo kaina auga?



Kas sudaro testavimo procesą?



Kokybės užtikrinimas ir kokybės kontrolė

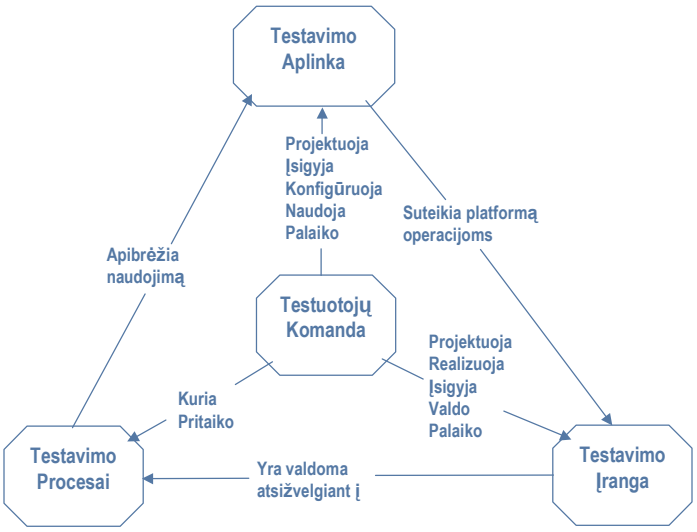
Kokybės kontrolė

- Nukreipta į konkretų produktą
- Tikrina ar produktas atitinka reikalavimus
- Indentifikuoja defektus, visų pirma tam, kad juos ištaisyti

Kokybės užtikrinimas

- Rūpinasi visais produktais, kurie bus kuriami pagal procesą
- Daro viską, kad išvengti defektų
- Kuria procesus
- Nustato kaip bus vertinami procesai
- Nustato procesų silpnąsias vietas ir jas gerina

Testavimo sistemos architektūra



Grafikas 1 Testavimo sistemos sandara

Testavimo planavimas

Testavimo apimties nustatymas

Klausimai testavimo apimčiai nustatyti:

- Ką mes galėtume testuoti?
- Ką mes turėtume testuoti?
- Ką mes galime testuoti?

Ką mes galėtume testuoti?

Du požiūriai į testavimą:

- Skaldantis – testavimo apimtis priklauso nuo testavimo tikslų:
  - Smulkus testavimo atvejis leidžia patikrinti žemo lygio detales
  - Stambus testavimo atvejis suteikia informacijos apie bendrą sistemos elgesį
- Fazinis – testavimas atliekamas įvairių testavimo fazių metu

Ką mes turėtume testuoti?

- Testavimo procesas leidžia įvertinti kokybės rizikas ir nustatyti defektų tipus, esančius testuojamoje sistemoje
- Kokybės patirtis yra vartotojų įspūdžiai apie tai, kaip gerai produktas atitinka jų lūkesčius
- Testavimo sistemos tikslumas parodo, kiek testavimo sistema leidžia iš anksto įvertinti vartotojų kokybės patirtis

Testavimo planas

Kuriami dokumentų tipai:

- Testavimo planas
- Testavimo atvejų dokumentai

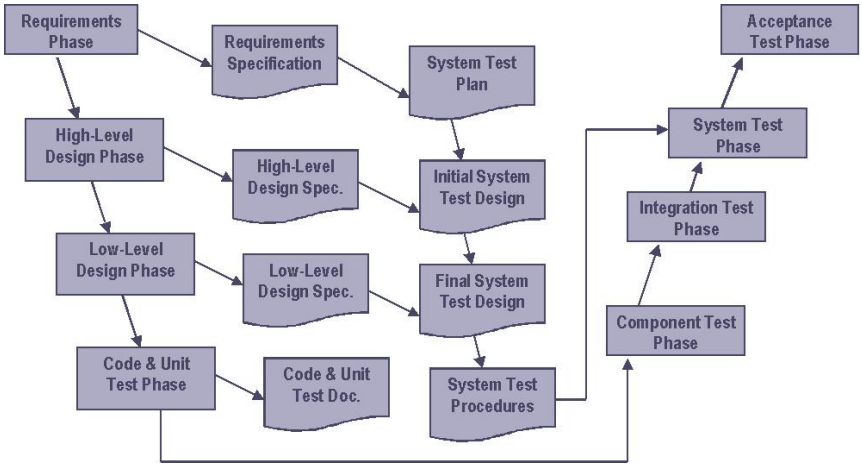
Testavimo planas atspindi testavimo strategiją, o testavimo atvejų rinkinys - taktiką

Testavimo plano privalumai:

- Galimybė surinkti mintis ir idėjas
- Komunikacijos su testuotojų ir programuotojų komandomis, o taip pat projekto vadovu gerinimas

PĮ Testavimo Etapai

V diagrama



V diagrama yra PĮ kūrimo ir testavimo modelis, parodantys sąsajas tarp PĮ kūrimo ir testavimo procesų

Komponento testavimas

Komponento testavimo tikslas – patikrinti, ar programos komponentas atitinka reikalavimus

Komponento testavimo taikymo sritys:

- Komponento atliekamų skaičiavimų teisingumas
- Žemo lygio našumo tikrinimas
- Žemo lygio patikimumo tikrinimas
- Lango turinys ir navigacija
- Bylų ar įrašų kūrimas, pakeitimas ir ištrynimas

Komponento testavimą paprastai atlieka programuotojų komanda

Integracijos testavimas

Integracijos testavimo tikslas – patikrinti, ar sudarančių sistemą modulių sąveika yra teisinga, stabili ir rišli

Integracijos testavimo taikymo sritys:

- Vieno modulio iškvietimas iš kito modulio
- Teisingas duomenų perdavimas tarp sąveikaujančių modulių
- Modulių suderinamumas
- Nefunkciniai klausimai tokie kaip sąsajos tarp modulių patikimumas

Integracijos testavimą paprastai atlieka programuotojų komanda

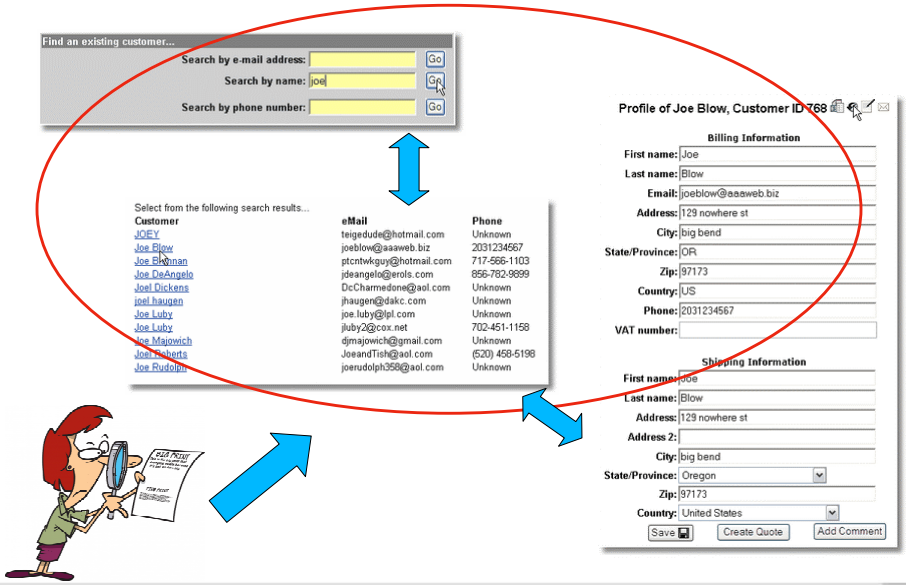
Sistemos testavimas

- Sistemos testavimo tikslas – įvertinti, ar sistema bus priimta vartotojų ir išvengti klaidų, galinčių turėti neigiamą įtaką vartotojo funkcinei, finansinei veiklai ar įvaizdžiui
- Sistemos testavimą paprastai atlieka testuotojų komanda

Patvirtinimo testavimas

- Patvirtinimo testavimo tikslas – įvertinti, ar sistema atitinka veiklos reikalavimus ir patikrinti, ar sistema veikia teisingai ir yra patogi, prieš ją formaliai atiduodant vartotojui
- Patvirtinimo testavimą paprastai atlieka vienas ar keli vartotojų atstovai padedami testuotojų komandos

Testavimo Etapai. Pavyzdys



Pakartotinis testavimas

- Pakartotinio testavimo tikslas – patikrinti, ar sistemoje neatsirado funkcinių ir nefunkcinių klaidų, susijusių su naujų funkcijų pridėjimu ar egzistuojančių funkcijų pokyčiais
- Pakartotinis testavimas yra daugiau testavimo technika nei testavimo etapas ir gali būti vykdomas visuose testavimo etapuose
- Pakartotinis testavimas dažniausiai taikomas sistemos ir patvirtinimo testavimo metu

Pakartotinis testavimas: taikymo metodai

Testavimo rinkinio pasirinkimo metodai:

- Prioritetų skyrimas – iš anksto kiekvienam testavimo rinkiniui yra priskiriamas prioritetas ir vėliau testavimas vykdomas, atsižvelgiant į testavimo rinkinių prioritetus
- Dinaminis prioritetų skyrimas – kiekvienam testavimo rinkiniui prioritetas yra priskiriamas kiekvieno testavimo ciklo pradžioje ir tada testavimas vykdomas pagal testavimo rinkinių prioritetus
- Šaudymas – atsitiktinis testavimo rinkinių paskirstymas tarp testavimo ciklų
- Geležinkelis – visų testavimo rinkinių vykdymas kiek įmanoma daugiau kartų

Testavimo Technikos

Testavimo technika ir įrankis – kur skirtumas?

- Įrankis yra priemonė testavimo procesui atlikti. Pavyzdžiui, plaktukas
- Tam, kad atlikti testavimą, vien tik įrankio nepakanka. Pavyzdžiui, kol nebus žinoma technika, kaip naudotis plaktuku, įrankis nebus naudojamas
- Testavimo technika yra procesas, skirtas užtikrinti, kad tam tikras sistemos ar jos vieno modulio aspektas veikia tinkamai
- Technikų yra keletas, o įrankių daug

Testavimo technikų kategorijos

- Bendros testavimo technikos – aukšto lygio požiūris į testavimą, kuris gali būti pritaikomas konkrečiom testavimo technikom apibūdintom sekančiose dviejose kategorijose
- Funkcinės testavimo technikos – naudojamos patikrinti, ar sistema atitinka funkcinius reikalavimus
- Nefunkcinės testavimo technikos – naudojamos patikrinti, ar sistema atitinka nefunkcinius reikalavimus

Bendros testavimo technikos

Bendros testavimo technikos apima:

- Teigiamą ir neigiamą testavimą
- Baltos ir juodos dėžės testavimą
- Tiriamąjį testavimą
- Automatizuotą PĮ testavimą

Teigiamas ir neigiamas testavimas

- Teigiamo testavimo tikslas – patikrinti, ar sistema daro tai, kas nurodyta reikalavimuose
- Neigiamo testavimo tikslas – parodyti, kad sistema nedaro to, ko ji neturėtų daryti
- Neigiamas testavimas dažnai naudojamas neaprašytų arba prastai dokumentuotų sistemos dalių patikrinimui

Baltos ir juodos dėžės testavimas

- Baltos dėžės testai yra kuriami ir atliekami remiantis žiniomis apie vidinę PĮ struktūrą (vidinis vaizdas)
- Juodos dėžės testai yra kuriami ir atliekami remiantis apibrėžtomis PĮ funkcijomis (išorinis vaizdas)
- Testavimo procese turėtų būti naudojamos abi technikos:
  - Baltos dėžės testavimas paprastai yra atliekamas ankstyvosiose testavimo proceso fazėse
  - Juodos dėžės testavimas paprastai yra atliekamas vėlesniuose testavimo proceso etapuose

Tiriamasis testavimas

- Tiriamasis testavimas – tai klaidų paieška remiantis intuicija ir patirtimi
- Atlikti tiriamąjį testavimą padeda:
  - Sistemos išmanymas
  - Ankstesnių testavimo etapų rezultatų žinojimas
  - Patirtis testuojant panašias sistemas
  - Dažniausiai pasitaikančių klaidų, tokių kaip dalyba iš nulio, žinojimas

Automatizuotas PĮ testavimas

- Automatizavimo tikslas turi būti suderintas su sistemos kokybės tikslais
- Testavimas automatizuojamas ne testavimo kaštų sumažinimui, o PĮ kūrimo proceso lankstumo padidinimui
- Automatizavimui turi būti taikomi PĮ kūrimo procesai
- Testavimo automatizavimas – didelė investicija

Funkcinės testavimo technikos

Funkcines testavimo technikos apima:

- Ekvivalentinio dalinimo testavimą
- Ribinių verčių analizę
- Įterpimo testavimą
- Perėjimo tarp būsenų testavimą
- Scenarijų testavimą

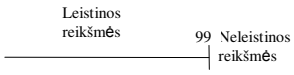
Ekvivalentinio dalinimo testavimas

- Ekvivalentinio dalinimo testavimo tikslas – minimizuoti testavimo atvejų skaičių, reikalingų įėjimo ir išėjimo sąlygų padengimui
- Technikos taikymas:
  - PĮ įėjimo ir išėjimo aibė padalinama į susijusias duomenų grupes vadinamas įėjimo/išėjimo sąlygomis
  - Kiekvienai įėjimo/išėjimo sąlygai nustatoma ekvivalentiška klasė (EK)
  - Iš kiekvienos EK parenkama reikšmė
  - Vieno EK duomenų pavyzdžio testavimas yra ekvivalentiškas visų EK duomenų testavimui

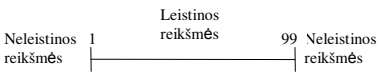
Ekvivalentinio dalinimo testavimas (tęs.)

Reikalingų EK kiekis priklauso nuo įėjimo/išėjimo sąlygos tipo. Jei įėjimo/išėjimo sąlyga yra:

- Reikšmių aibė → 1 EK aibėje, po 1 EK už aibės ribų (pvz. Visos reikšmės iki 99 imtinai)



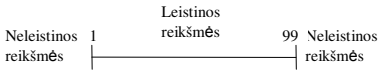
- Rinkinys reikšmių → 1 EK teisingoms reikšmėms, 2 EK neteisingoms reikšmėms (pvz. reikšmė intervale [1;99])



- "Turi būti" *situacija* → 1 EK teisingai reikšmei, 1 EK neteisingai reikšmei

Ribinių verčių analizė

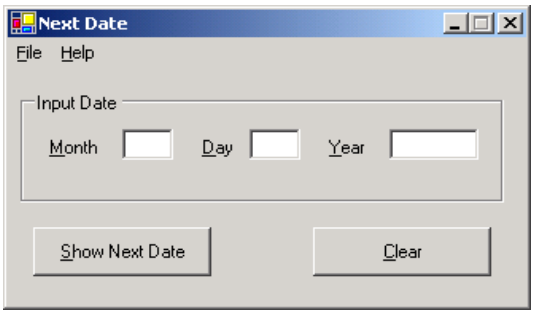
- Ribinių verčių analizės technika papildo ekvivalentinį dalinimą
- Ribinių verčių analizės tikslas – patikrinti ribines vertes
- Technikos taikymas:
  - Kiekvienoje EK parenkama ribinė reikšmė, reikšmė prieš ribinę reikšmę ir reikšmė po ribinės reikšmės



  - Ribinės reikšmės intervalo [1;99] būtų {0,1,99,100}

Ribinių verčių analizė. Pavyzdys

Programa grąžinanti sekančią kalendorinę dieną formatu mėnuo/diena/metai



Ribinių verčių analizė. Pavyzdys

Bendra formulė:

(6*n + 1)=T

n – kintamųjų kiekis

T – testavimo atvejai

(6*3 + 1)=19

Kai kiekviena reikšmė laikoma nepriklausoma

Test	Month	Day	Year	Expected Result
1	**0** (min -1)	**15**	**1959**	**Error**
2	**1** (min)	**15**	**1959**	**1/16/1959**
3	**2** (min + 1)	**15**	**1959**	**2/16/1959**
4	**11** (max -1)	**15**	**1959**	**11/16/1959**
5	**12** (max)	**15**	**1959**	**12/16/1959**
6	**13** (max +1)	**15**	**1959**	**Error**
7	**6**	**0** (min -1)	**1959**	**Error**
8	**6**	**1** (min)	**1959**	**6/2/1959**
9	**6**	**2** (min + 1)	**1959**	**6/3/1959**
10	**5**	**30** (max -1)	**1959**	**5/31/1959**
11	**5**	**31** (max)	**1959**	**6/1/1959**
12	**5**	**32** (max +1)	**1959**	**Error**
13	**6**	**15**	**1581** (min -1)	**Error**
14	**6**	**15**	**1582** (min)	**6/16/1582**
15	**6**	**15**	**1583** (min + 1)	**6/16/1583**
16	**6**	**15**	**2999** (max -1)	**6/16/2999**
17	**6**	**15**	**3000** (max)	**6/16/3000**
18	**6**	**15**	**3001** (max +1)	**Error**
19	**6** (nom)	**15** (nom)	**1918** (nom)	**6/16/1918**

Ribinių verčių analizė. Pavyzdys

Dienos reikšmė turi 3 priklausomybes:

- 30 dienų per mėnesį
- Vasaris lyginiais metais
- Vasaris nelyginiais metais

Test	Month	Day	Year	Expected Result
20	6	29	1959	6/30/1959
21	6	30	1959	7/1/1959
22	6	31	1959	Error
23	2	27	1995	2/28/1995
24	2	28	1995	3/1/1995
25	2	29	1995	Error
26	2	28	1996	2/29/1996
27	2	29	1996	3/1/1996
28	2	30	1996	Error

Įterpimo testavimas

- Įterpimo testavimas – tai pakeitimų programos kode darymas, siekiant geriau atlikti testavimą
- Tai rizikinga technika, nes priduota sistema nėra ta, kuri buvo iš tikro testuota

Perėjimo tarp būsenų testavimas

- Perėjimo tarp būsenų testavimo technikos tikslas yra sistemos analizė atsižvelgiant į:
  - būsenas, kuriose sistema gali būti
  - perėjimus tarp būsenų
  - veiksmus, kurie sukelia perėjimus
  - veiksmus, kuriuos iššaukia perėjimai
- Technika ypač naudinga reikalavimų trūkumams nustatyti

Scenarijų testavimas

- Scenarijų testavimas naudojamas veiklos reikalavimų patikrinimui
- Technika apima tris atributus:
  - Testas turi būti realus
  - Testas turi būti sudėtingas, apimantis keletą funkcijų
  - Nustatymas, ar testas suveikė ar ne, turi būti greitas ir lengvas
- Daugiau scenarijų → išsamesnis testavimas
- Technika naudojama vėlesniuose testavimo proceso etapuose

Nefunkcinės testavimo technikos

- Konfigūracijos testavimas – įvertina, ar sistema veikia tinkamai su reikiamomis PĮ konfigūracijomis
- Suderinamumo testavimas - patvirtina, kad sistema nedaro žalos kitoms suinstaliuotoms sistemoms ir atvirkščiai
- Tarpusavio sąveikos testavimas - patikrina, ar sistema gali sėkmingai komunikuoti su kitomis sistemos, nuo kurių ji priklauso
- Dokumentacijos ir pagalbos sistemos testavimas - patikrina vartotojo dokumentacijos tikslumą, užbaigtumą ir vartojimo patogumą

Nefunkcinės testavimo technikos (tęs.)

- Apsaugos testavimas:
  - Funkcijų lygmenyje – ar vartotojui prieinamos tik tos funkcijos ir duomenys, kurie yra leidžiami tam vartotojo tipui
  - Sistemos lygmenyje – ar sistema prieinama tik tiems vartotojams, kuriems suteiktos tokios teisės
- Patogumo testavimas – patikrina, ar sistemos vartotojai norimus veiksmus gali atlikti efektyviai
- Diegimo testavimas: 1) patikrina, ar PĮ gali būti įdiegta tiek normaliomis, tiek nenormaliomis sąlygomis; 2) įvertina, ar įdiegta PĮ funkcionuoja teisingai; 3) patvirtina, kad PĮ gali būti ne tik įdiegta, bet ir pašalinta

Nefunkcinės testavimo technikos (tęs.)

- Našumo testavimas – atranda sistemos komponentų našumo trūkumus, ištiria sistemą, jai veikiant nustatytą laiko tarpą su įvairiu apkrovimu
- Patikimumo testavimas – užtikrina, kad sistema funkcionuoja apibrėžtą laiko tarpą
- Atsistatymo po klaidos testavimas – patikrina, ar 1) sistemos atsistatymas vyksta sklandžiai; 2) atsistatymo procesas yra efektyvus
- Stresinis testavimas – patikrina sistemą stresinėmis sąlygomis, kai sistemai yra reikalingas nenormalus resursų kiekis, dažnumas ar dydis
- Apimties testavimas – įvertina sistemos galimybes apdoroti didelius duomenų kiekius

Testavimo projektavimas

Kaip projektuojamas testavimas?

1 žingsnis: Išskiriami testavimo scenarijai

Vartotojas gali susikurti e-pašto dėžutę, kurios adresas dar nėra registruotas.

1. Vartotojas nurodo jau registruotą e-pašto dėžutės adresą.
1. Vartotojas nurodo dar neregistruotą e-pašto dėžutės adresą.
1. Vartotojas nenurodo e-pašto dėžutės adreso.

Kaip projektuojamas testavimas?

2 žingsnis: Išskiriami testavimo atvejai

Vartotojas nurodo dar neregistruotą e-pašto dėžutės adresą.

1. Adresas nurodytas teisingai
1. Adresas nurodytas neteisingai
1. Adresas įvestas didžiosiomis raidėmis

Kaip projektuojamas testavimas?

3 žingsnis: Išskiriamos testavimo procedūros

Įvestas e-pašto adresas (vardo dalis iki @ simbolio) gali būti sudarytas tik iš lotyniškų raidžių, gali turėti simbolius . ir _ tačiau tai neturi būti pirmas ar paskutinis pavadinimo simbolis. Vardas gali turėti skaitmenis.

1. Adresas turi tarpą
1. Adresas turi neleistinų simbolių (pvz. / ; : “ ,)
1. Vardas turi leistiną simbolį.
1. Adresas sudarytas iš nelotyniškų raidžių (pvz. a, č, š, ž)
2. Adreso pradžioje/gale nurodytas _ arba .

Kaip funkcinį testavimą projektuojame mes?

Testo ID ir numeris. Testo ID yra FT<numeris>

Tikslas. Trumpas testo tikslo aprašymas

Išankstinės sąlygos. Jei yra, sąrašas išankstinių sąlygų

Susiję reikalavimai. Sąrašas reikalavimų, susijusių su testu

Aktoriai. Sąrašas aktorių, galinčių vykdyti testą

Testavimo scenarijai.

#	Testavimo atvejis	Laukiami rezultatai	Testavimo procedūros	Taip/ ne
1.				
2.				
3.				
4.				

Kaip funkcinį testavimą projektuojame mes?

ID	Testavimo atvejis	Laukiami rezultatai	Testavimo procedūros	Taip/Ne
1.	Vartotojas nurodo jau registruotą e-pašto dėžutės adresą.	Išmetamas pranešimas „Tokia pašto dėžutė jau užregistruota“		
2.	Vartotojas neteisingai nurodo dar neregistruotą e-pašto dėžutės adresą.	Išmetamas pranešimas „Neteisingai nurodytas adresas“. Adreso įvedimo laukas išvalomas ir į jį pastatomas žymeklis.	1. Adresas turi tarpą 2. Vardas prasideda tašku 3. Vardas prasideda _ 4. Vardas baigiasi . 5. Vardas baigiasi _ 6. Adresas turi neleistinų simbolių (pvz. / ; : “ ,) 7. Adresas sudarytas iš nelotyniškų raidžių (pvz. a, č, š, ž)	
3.	Vartotojas teisingai nurodo dar neregistruotą e-pašto dėžutės adresą.	Išmetamas pranešimas „Jūsų pašto dėžutė sėkmingai sukurta“.	1. Vardas neturi kitų simbolių tik lotyniškas raides 6. Vardas turi skaitmenį	
4.				

Defektų valdymas

Defektų ataskaitos

- Defekto ataskaita yra techninis dokumentas, aprašantis su viena klaida susijusius sistemos gedimo simptomus
- Defekto ataskaita yra ne tik aprašomi klaidos simptomai, bet ir:
  - Įvertinama rizika, kad klaidą pastebės tipinis sistemos vartotojas,
  - Nurodoma žala, kurią klaida daro vartotojui ir sistemai,
  - Izoliuojant defektą padedama tiksliau nustatyti klaidos ištaisymo kaštus ir riziką.

Defektų raportavimo procesas

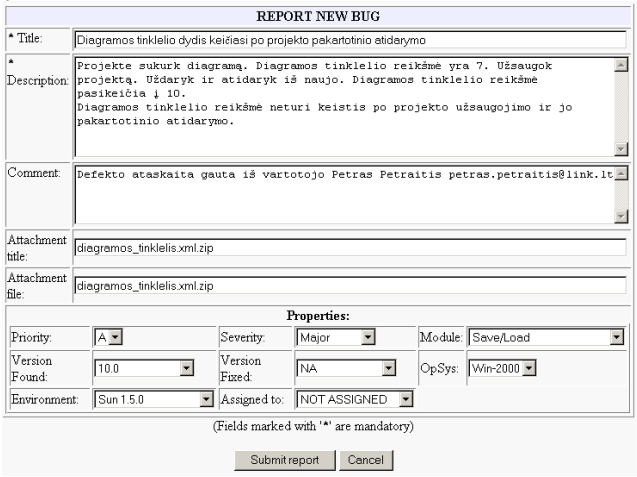
1. Pakartojimas - prieš rašant defekto ataskaitą, klaidinga situacija turi būti pakartota bent keletą kartų
2. Perteklinių žingsnių pašalinimas - defekto ataskaitoje turi būti palikti tik tie žingsniai, kurie yra tiesiogiai susiję su klaida.
3. Apibendrinimas - defekto apibendrinimas padeda tiksliau įvertinti klaidos poveikį sistemą ir įvertinti jos ištaisymo svarbą
4. Dviprasmiškumo panaikinimas - defekto ataskaitoje neturi būti vartojami žodžiai ar frazės, kuriuos perskaičius kiltų klausimas: ką tiksliai kokybės inžinierius norėjo pasakyti

Defektų raportavimo procesas (tęs.)

1. Defekto prioriteto ir poveikio įvertinimas:
   - Prioritetas - tikimybė, kad sistemos tipinis vartotojas pastebės defektą
   - Poveikis – žala, kurią defektas daro sistemai ir vartotojui
     - Kritinis. Paprastai reiškia sistemos “nulūžimą” ir didžiulę grėsmę sistemos stabilumui
     - Stabdantis. Kol klaida nebus ištaisyta, tolimesnis modulio ar funkcijos testavimas yra negalimas
     - Labai svarbus. Klaidą apeiti galima, tačiau apėjimo kelias sudėtingas
     - Nedidelis. Klaidą ištaisyti reikia, tačiau pati klaida nedaro didelės žalos. Klaidos apėjimo kelias yra lengvas arba jo nereikia iš vis
     - Kosmetinis. Paprastai naudojama vartotojo sąsajos klaidoms
     - Pasiūlymas. Modulio ar funkcijos pagerinimo idėja

Defektų raportavimo procesas (tęs.)

1. Papildoma informacija - kas rado defektą, kada jis buvo rastas, kokiame sistemos modulyje, kokioje testuojamos sistemos versijoje ir pan.



REPORT NEW BUG

* Title:	Diagramos tinklelio dydis keičiasi po projekto pakartotinio atidarymo
* Description:	Projekte sukurk diagramą. Diagramos tinklelio reikšmė yra 7. Užsaugok projektą. Uždaryk ir atidaryk iš naujo. Diagramos tinklelio reikšmė pasikeičia į 10. Diagramos tinklelio reikšmė neturi keistis po projekto užsaugojimo ir jo pakartotinio atidarymo.
Comment:	Defekto ataskaita gauta iš vartotojo Petras Petraitis petras.petraitis@link.lt
Attachment title:	diagramos_tinklelis.xml.zip
Attachment file:	diagramos_tinklelis.xml.zip

Properties:

Priority:	A	Severity:	Major	Module:	Save/Load
Version Found:	10.0	Version Fixed:	NA	OpSys:	Win-2000
Environment:	Sun 1.5.0	Assigned to:	NOT ASSIGNED		

(Fields marked with '*' are mandatory)

Submit report | Cancel

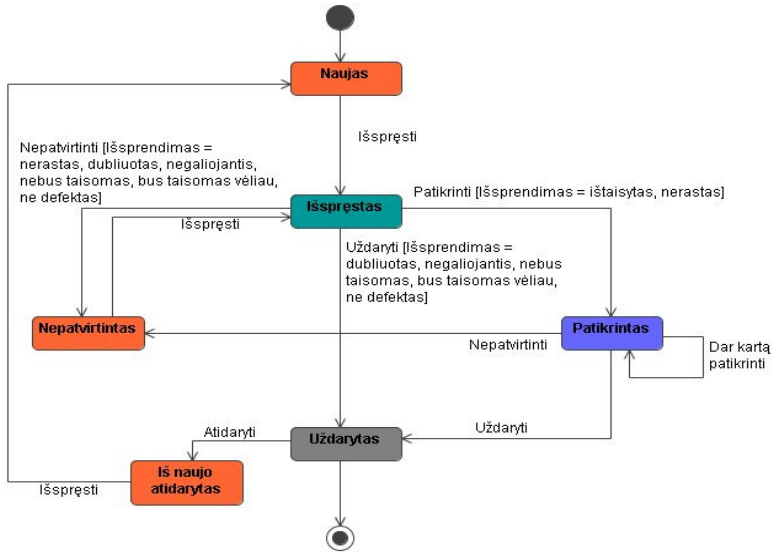
Defektų raportavimo procesas (tęs.)

1. Santrauka - trumpas klaidos apibūdinimas vienu sakiniu
2. Peržiūra - padeda anksti atrasti defektus ir taip sumažinti jų ištaisymo kaštus

Defektų valdymo sistemos

- Defektų valdymo sistema yra programa, suteikianti projekto komandai galimybes kurti, valdyti ir analizuoti defektų ataskaitas bei defektų tendencijas
- Pagrindinis defektų valdymo sistemos tikslas – užtikrinti, kad defektai, kurie turi būti ištaisyti, būtų pašalinti
  - Apie problemą yra pranešama iš kart po defekto ataskaitos sukūrimo
  - Nėra defektų, kuriuos programuotojas pamiršo arba nenorėjo taisyti
  - Defektų, neištaisytų dėl prasto raportavimo, skaičius yra minimalus

Defektų būsenų valdymas



Grafikas4 Defektų būsenų diagrama

Defektų būsenų valdymas(tęs.)

Defektą apibūdina būsena, išsprendimas, svarumas

Būsenos reikšmės:

- *Naujas* – naujai įrašytas defektas
- *Išspręstas* – defektą peržiūrėjo ir išsprendė programuotojas
- *Nepriimtas* – defekto išsprendimas neteisingas
- *Patikrintas* – defekto išsprendimas teisingas
- *Užšaldytas* – defektą sunku pakartoti, todėl jis negali būti ištaisytas
- *Uždarytas* – paskutinė defekto gyvavimo ciklo būsena
- *Iš naujo atidarytas* – defektas atidarytas, nes iš naujo atsirado arba buvo nuspręsta jį taisyti

Svarumo reikšmės:

- *Vieną kartą testuotas*: defekto išsprendimas tikrintas vieną kartą. Defekto išsprendimas patvirtintas.
- *Pakartotinai testuotas*: defekto išsprendimas pakartotinai tikrintas. Defekto išsprendimas patvirtintas.

Defektų būsenų valdymas(tęs.)

Išsprendimo reikšmės:

- *Ištaisytas*: programuotojas ištaisė defektą
- *Negaliojantis*: defekto aprašymas neatitinka dabartinės situacijos
- *Nebus taisomas*: defektas nebus taisomas
- *Bus taisomas vėliau*: defekto ištaisymas yra nukeliamas nurodytam laiko tarpui
- *Dubliuotas*: toks defektas jau yra užregistruotas defektų valdymo sistemoje
- *Ne defektas*: testuotojas klaidingai suprato funkcionalumą
- *Nerastas*: pagal pateiktą defekto aprašymą programuotojas nepakartojo defekto
- *Atmestas*: testuotojas nepatvirtino defekto išsprendimo
- *Neištaisytas*: nors defekto išsprendimas yra *ištaisytas*, defektas egzistuoja; jei defekto išsprendimas *nerastas* – testuotojas pakartojo klaidą

Defektų analizė, naudojant kokybės metrikas

PĮ defektų metrikos

- Metrika – kiekybinis matas, parodantis, kiek sistema ar jos komponentas atitinka nustatytus atributus
- Projektuose gali būti naudojamos standartinės PĮ metrikos arba sukurti individualūs, pritaikyti projektui arba įmonės procesui matavimai

PĮ defektų metrikos

- **Naujų ir ištaisytų defektų diagrama**
- **Defektų atsiradimo priežasčių diagrama**
- **Defektų ištaisymo periodo diagrama**
- **Posistemių diagrama**
- **Defektų radimo procentinis matas**

Naujų ir ištaisytų defektų diagrama

Grafikas parodo santykinį dydį tarp naujai įrašytų defektų sukauptos sumos bei ištaisytų defektų sukauptos sumos. Dydžiai skaičiuojami kiekvieną dieną.

Ištaisytų ir naujų defektų santykio metrika atsako į klausimus:

- ar jau galima atiduoti produktą?
- ar programuotojai jau baigia taisyti defektus?
- ar defektų valdymo procesai veikia?
- kaip projekto etapai veikia defektų atsiradimą arba ištaisymą?

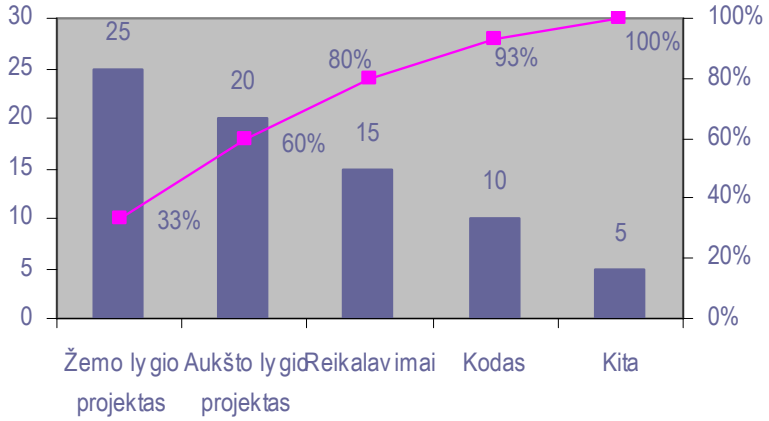
Defektų atsiradimo priežasčių diagrama

Atsiradimo priežasčių diagrama rodo kiekvienos klaidos tipo įtaką visiems rastiems ir ištaisytiems defektams.

Rinkti klaidų priežasčių duomenis yra naudinga:

- parodo kūrimo komandai bei vadovams vietas, kurios labiausiai įtakoja klaidų atsiradimą
- ilgame periode, ši informacija gali būti naudojama proceso peržiūrai ir gerinimui

Pavyzdys – klaidų įtaka



Paveikslas 6. Klaidų priežasčių diagrama

Defektų ištaisymo periodo diagrama

- Defektų ištaisymo periodo diagrama parodo kasdienį ir svyruojantį defektų ištaisymo periodą.
- Diagrama parodo kaip greitai programuotojai atsiliepia į testuotojų defektų raportus.
- Kasdienis ištaisymo periodas - tai vidutinis dienų skaičius tarp defekto įrašymo ir jo ištaisymo. Reikšmė skaičiuojama kiekvieną dieną.
- Svyruojantis ištaisymo periodas - tai vidutinis dienų skaičius visiems ištaisytiems defektams, įtraukiant šios dienos ir visų ankstesnių dienų defektus.

Defektų ištaisymo periodo diagrama

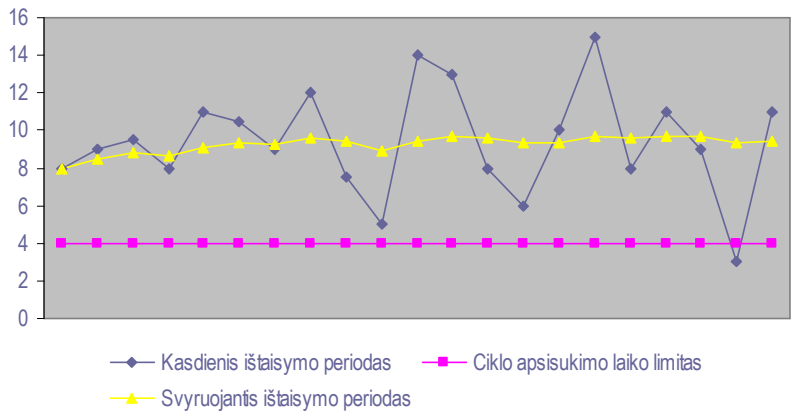
Defektų ištaisymo periodas gali būti traktuojamas kaip stabilumas bei priimtinumas:

- stabilaus periodo grafikas rodo santykinai mažą skirtumą tarp kelių dienų, su besileidžiančia svyruojančio ištaisymo periodo grafiko kreive, dažnai artėjančia prie 0

- priimtino periodo grafike kasdienio periodo ir svyruojančio periodo grafikai patenka tarp viršutinių ir apatinių limito rėžių, nustatytų projekte, testavimo plane arba defekto gyvavimo periode

Defektų ištaisymo periodo grafikas, kuris yra ir stabilus, ir priimtinas, rodo gerai suprantamą ir sklandžiai funkcionuojantį defektų valdymo procesą

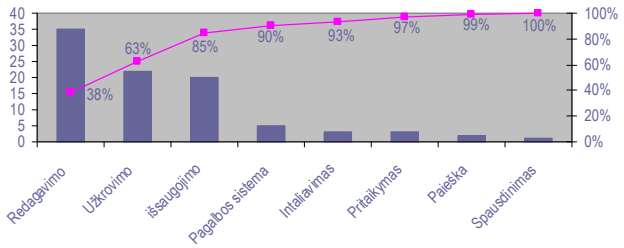
Pavyzdys – defektų ištaisymo periodas



Paveikslas 7. Defektų ištaisymo periodo grafikas

Posistemių diagrama

- Posistemių diagrama parodo, kuriose posistemėse yra daugiausiai defektų.
- Posistemių diagrama gali būti naudojama nustatant produkto bei proceso gerinimo pastangas.
- Prieš nusprendžiant, kurias posistemes testuoti, turi būti nustatomos vartotojui svarbios kokybės rizikos.



Paveikslas 8. Posistemių išskaidymo grafikas

Defektų radimo procentinis matas

Testavimo metu praslydę defektai yra tokie, kurių testavimo komanda nerado, bet įvertinus realias priežastis galėjo rasti testavimo metu.

Testavimo metu praslydę defektai paprastai iškyla vienoje arba keliose situacijose:

- mažo patikimumo testavimo sistema – testavimo sistema neįtraukia tų dalių, kurios yra ypač svarbios vartotojui
- regresijos testavimo "skylės" – testavimo atvejai yra praleidžiami prieš klaidos patekimą į sistemą
- testavimo rezultatų interpretavimo klaida – testuotojas nepakankamai įvertino problemos aktualumą

Defektų radimo procentinis matas

Testavimo metu praslydę defektai yra apskaičiuojami naudojant defektų radimo procentinio mato metriką

$$DDP = \frac{bugs_{testers}}{bugs_{testers} + bugs_{customers}} * 100\%$$

Metrikai apskaičiuoti yra pasirenkamas atitinkamas periodas po produkto išleidimo.

Bendri pastebėjimai metrikoms

- Naudokite grafikus bei metrikas įvairiems projektų palyginimams.
- Nesiremkite grafikais aklai.
- Defektų bei testavimo duomenų analizė duoda realią reikšmę tik tada, kai esminiai duomenys yra atidžiai surinkti, pilni ir neklastojami.
- Kad gautumėte tikslesnius įvertinimus, naudokite ir kitokias metrikas, skirtas testavimo bei kitų projekto dalių, proceso ar pačio projekto valdymui.

Programinės įrangos kokybės metrikos

Dažniausiai naudojamų metrikų pavyzdžiai ir apibrėžimai

PĮ metrikų kategorijos

Programinės įrangos metrikos gali būti skirstomos į 3 kategorijas:

- Produkto metrikos, nusakančios kokybines produkto charakteristikas;
- Proceso metrikos, naudojamos programinės įrangos kūrimo ir palaikymo procese;
- Projekto metrikos, nusakančios projekto kokybines charakteristikas ir būseną.

Proceso metrikos yra strateginės reikšmės, o projekto metrikos – taktinės reikšmės.

Programinės įrangos kokybės metrikos

Kokybinės PĮ metrikos gali būti skirstomos:

- Galutinio produkto kokybės metrikos,
- Esamo kūrimo momento (*in-process*) kokybės metrikos,
- PĮ palaikymo metrikos.

Programinės įrangos inžinerijos uždavinys yra išanalizuoti ryšius tarp esamo kūrimo momento kokybės metrikų, projekto charakteristikų ir galutinio produkto kokybės ir pagal rezultatus pagerinti proceso ir produkto kokybę.

Galutinio produkto kokybės metrikos

Galutinio produkto kokybės metrikų pavyzdžiai:

- Vidutinis laiko tarpas tarp "lūžimų" (*MTTF – mean time to failure*),
- Defektų tankis,
- Vartotojų problemų metrikos,
- Vartotojų pasitenkinimo metrikos.

MTTF metrika

MTTF metrika nusako laiką tarp dviejų klaidų ar lūžimų, arba kaip ilgai vidutiniškai programinė įranga gali veikti be klaidų;

Ši metrika dažniausiai naudojama kritinio saugumo sistemose;

MTTF metrika paprastai yra naudinga sistemose, kuriose yra naudojamos ilgos transakcijos;

Tačiau ši metrika nėra dažnai naudojama kuriant komercinės paskirties programinės įrangos produktus, nes:

- Yra sunku surinkti duomenis, jeigu MTTF yra ilgas;
- Dažnai yra sudėtinga arba neįmanoma automatiškai užregistruoti kai kurių lūžimų laiko;
- Laiko tarp lūžimų duomenys turi būti tikslūs;
- MTTF gali būti skirtingas dėl vykdymo aplinkos konfigūracijos.

Defektų tankio metrika

Defektų tankio metrika apibrėžiama kaip defektų kiekio santykis su programinės įrangos produkto apimtimi:

$$\text{Defektų tankis} = \frac{\text{Defektų skaičius}}{\text{PĮ apimtis}}$$

Defektų kiekio matu laikomas per tam tikrą laikotarpį atrastų defektų kiekis, pvz. rastų defektų skaičius nuo modulio sukūrimo iki esamos datos.

Dydis dažniausiai yra apibrėžiamas tūkstančiais kodo eilučių (*KLOC – Kilo Lines of Code*).

Defektų tankio metrikos taikymas

Defektų tankio metrika yra naudojama daugelyje komercinių programinės įrangos sistemų, nes yra naudingas rodiklis lyginant:

- Defektų tankius skirtinguose programinės įrangos komponentuose. Tai padeda identifikuoti kandidatus detalesnei peržiūrai, testavimui, struktūros pertvarkymui (*re-engineering*) ar pakeitimui;
- Programinės įrangos produkto versijas, sekant kokybės lygį ar vertinant defektų taisymo ar kokybės pagerinimo veiksmus. Kokybės skirtumai tarp skirtingų produktų ar produktų linijų taip pat gali būti įvertinti naudojant šią metriką.

✓ *Nereikia pamiršti, kad vertinant skirtingų produktų ar produkto versijų defektų tankius reikia atsižvelgti, ar buvo skirtas adekvatus testavimo laikas ir ar apimtis apibrėžiama adekvačiais matais, pvz. ar į kodo eilučių skaičių įtraukiami komentarai*

Defektų tankio metrikos taikymo pavyzdys

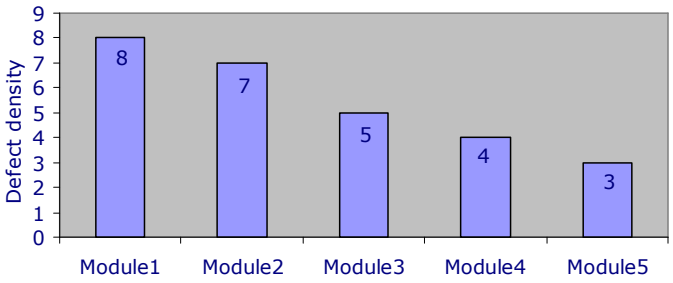


Figure 4 Relative number of defects in various modules

Klausimas: Kuriam moduliui reikėtų skirti daugiau testavimo?

Atsakymas: Pirmiausia reikėtų identifikuoti didžiausios rizikos modulius, o tik po to spręsti pagal defektų tankį.

Vartotojų problemų metrika

Vartotojų problemų metrika nurodo, kiek problemų vartotojai turi naudodami produktą.

Vartotojo požiūriu, visos problemos, su kuriomis jis susiduria naudodamas produktą – ne vien tiktai tikri defektai – yra programinės įrangos problemos.

Problemos, kurios nėra tikri defektai, yra vadinamos vartojimo problemomis, pvz. neaiški dokumentacija.

Vartojimo problemos kartu su tikrais defektai sudaro visą problemų aibę, žiūrint iš vartotojo perspektyvos.

Vartotojų problemų metrikos apibrėžimas

Klientų problemų metrika paprastai yra išreiškiama vidutiniu problemų kiekiu per vartotojo mėnesį (*PUM – problem per user month*):

$$PUM = \frac{\text{Bendras per laikotarpį klientų raportuotų problemų skaičius}}{\text{<Per laikotarpį instaliuotų licenzijų kiekis> × <mėnesių skaičius skaičiavimo laikotarpyje>}}$$

Vartotojų problemų metrikos taikymas

PUM paprastai yra skaičiuojama kiekvieną mėnesį po programinės įrangos produkto versijos išleidimo. Taip pat skaičiuojamas ir metinis PUM vidurkis.

Žemas PUM gali būti pasiekiamas:

- Pagerinant kūrimo procesą ir sumažinant produkto defektų;
- Sumažinant vartojimo problemų kiekį;
- Padidinant produkto pardavimus.

Vartotojų pasitenkinimo metrika

Vartotojų pasitenkinimas paprastai įvertinamas, vykdant apklausas ir naudojant tokius galimus atsakymus:

- Labai patenkintas;
- Patenkintas;
- Neutralus;
- Nepatenkintas;
- Labai nepatenkintas.

Vėliau pagal apklausos rezultatus yra skaičiuojamos įvairios metrikos:

- kiek % vartotojų patenkinti;
- kiek % vartotojų nepatenkinti;
- …

Proceso metrikos

Lyginant su galutinio produkto kokybės metrikomis, proceso kokybės metrikos yra mažiau formaliai apibrėžtos.

Tokių metrikų pavyzdžiai:

- Defektų tankio kitimas - ši metrika paprastai yra stebima, siekiant palyginti dvi to paties produkto versijas, kurias kūrė ta pati organizacija
- Defektų radimo pobūdis - įvertina, kiek defektų randama sistemoje per tam tikrą laiką
- Defektų pašalinimas pagal fazes - ši metrika parodo kūrimo proceso sugebėjimą pašalinti atrastus defektus. Defektų kiekio kitimas turi būti sekamas visose programinės įrangos kūrimo fazėse.

Pavyzdys: defektų tankio stebėjimas produkto versijose

Ar produkto kokybė versijoje N+1 pagerėjo?

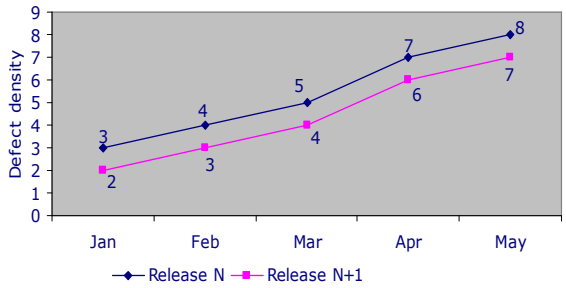


Figure 5 Defect density in releases N and N+1

Pavyzdys: defektų kiekio metrikų palyginimas

Bendras defektų kiekis abiem produktams yra vienodas, tačiau kokybė – skirtinga.

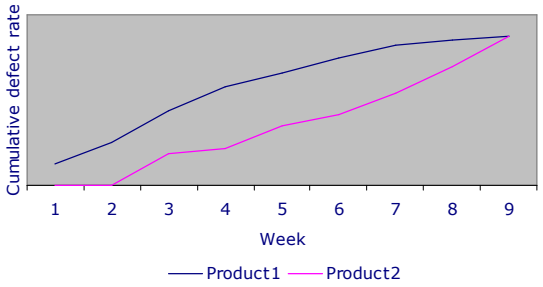


Figure 6 Cumulative defect rate for product1 and product2

Pavyzdys: defektų radimo metrikų palyginimas

Defektų radimo metrikos kitimas produktui 2 parodo, kad testavimas buvo nutrauktas per anksti.

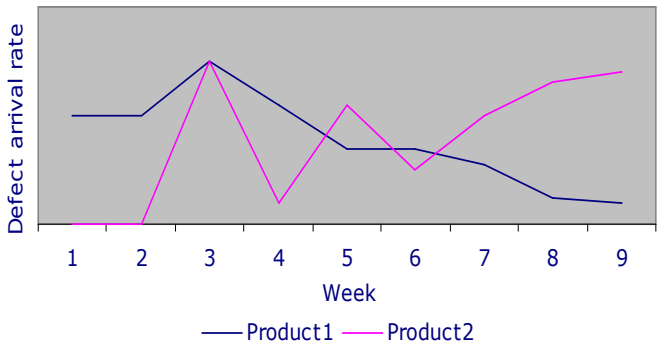


Figure 7 Defect arrival patterns for product1 and product2

Programinės įrangos palaikymo metrikos

Programinės įrangos produkto kokybė paprastai negali būti žymiai pakeičiama palaikymo fazėje.

Visgi palaikymo fazės tikslas yra padidinti vartotojų pasitenkinimą taisant rastus defektus kaip įmanoma greičiau ir kartu kokybiškai.

Palaikymo metrikų pavyzdžiai:

- Ištaisytų defektų kiekis;
- Defektų ištaisymo indeksas;
- Vidutinis ištaisymo laikas;
- Vėluojamų ištaisyti defektų procentas;
- Defektų ištaisymo kokybė.

Defektų ištaisymo indeksas

Defektų ištaisymo indeksas (*BMI – backlog management index)* nurodo, kiek ištaisytų defektų kiekio santykį su naujai užregistruotomis problemomis

BMI paprastai skaičiuojamas kiekvieno mėnesio ar savaitės pabaigoje.

$$BMI = \frac{\text{Per laikotarpį ištaisytų defektų kiekis}}{\text{Per laikotarpį užregistruotų defektų kiekis}} \times 100\%$$

- ✓ Naudojant tendencijas parodančius grafikus, neištaisytų defektų metrika gali pateikti naudingos informacijos palaikymo proceso valdymui;
- ✓ Ši metrika nurodo darbo krūvį programinės įrangos palaikymo fazėje;
- ✓ Reikia stengtis, kad BMI būtų didesnis už 100%, nes kitaip atvirų defektų kiekis vis auga.

Vidutinio ištaisymo laiko metrika

Nurodo vidutinį defekto ištaisymo laiką.

Paprastai kriterijus, per kiek laiko defektas turėtų būti ištaisytas priklauso nuo užregistruoto defekto žalingumo ir prioriteto.

Vėluojamų ištaisyti defektų procentas

Vėluojamų ištaisyti defektų procentas (*PDF – percent delinquent fixes*) skaičiuojamas pagal žemiau pateiktą formulę:

$$PDF = \frac{\text{Per nurodytą laikotarpį pavėluotai ištaisytų defektų kiekis}}{\text{Bendras per laikotarpį ištaisytų defektų kiekis}} * 100\%$$

✓Ši metrika negali būti naudojama realaus laiko vėluojamų ištaisyti defektų valdymui, kadangi jinai skaičiuojama, įtraukiant tiktai jau išspręstus defektus

Defektų ištaisymo kokybės metrika

Ši metrika skaičiuoja netinkamai ištaisytų defektų kiekį per tam tikrą laikotarpį.

Defektas yra netinkamai ištaisytas jeigu jisai neišsprendė nurodytos problemos, arba sąlygojo naujo defekto atsiradimą.

Netinkamai ištaisyto defekto vertinimas galimas dviem būdais:

- Laiko intervalu, kol jis buvo pastebėtas;
- Laiko intervalu, kol jis buvo tinkamai ištaisytas.

Pirmas būdas nurodo nuo vartotojo priklausomą metriką, antras – nuo proceso priklausomą metriką.

Pagrindinės taisyklės, kaip išvengti žmogiškojo faktoriaus

Neskaičiuokite metrikų individualiai darbuotojams

- Dėmesys turi būti kreipiamas į procesus bei produktus, ne į žmones

Niekada nenaudokite metrikos kaip "bausmės"

- Jeigu metrika remiamasi prieš žmogų ar grupę, galimas netinkamas duomenų rinkimas

Neignoruokite duomenų

- Jeigu vadovybės skelbiami tikslai neatitiks veiksmų, žmonės pradės dirbti pagal vadovybės veiksmus, o ne pagal tikslus

Niekada nenaudokite tiktai vienos metrikos

- Naudojant vieną metriką galima pagerinti vieną kokybės atributą kitų kaina

Pagrindinės taisyklės, kaip išvengti žmogiškojo faktoriaus

Parinkite metrikas pagal siekiamus tikslus

- Tiktai tos metrikos suteiks organizacijai reikalingą informaciją

Pateikite grįžtamąją informaciją

- Grįžtamosios informacijos pateikimas komandos nariams turi kelis privalumus:
  - Padeda išlaikyti dėmesį duomenų rinkimo poreikiui;
  - Komandos nariams kils mažiau įtarimų dėl neaiškaus metrikų panaudojimo;
  - Įtraukiant darbuotojus į duomenų analizę, organizacija gaus naudos iš jų žinių ir patirties;
  - Grįžtamosios informacijos apie duomenų rinkimo ir integralumo problemas pateikimas padės apmokyti darbuotojus tinkamai rinkti metrikų duomenis.