Proqram Təminatının Verifikasiya və Sınaq Metodlarının Müqayisəli Təhlili

Tamilla Bayramova[1], Nuranə Abbasova[2]
[1,2]AMEA İnformasiya Texnologiyaları İnstitutu, Bakı, Azərbaycan
[1]*tamilla@iit.ab.az*, [2]*nuraabbas.na@gmail.com*

***Xülasə*— İşdə proqram təminatının (PT) verifikasiya metodlarının klassifikasiyası və qısa xülasəsi verilmiş, verifikasiyanın məqsədləri və metodları araşdırılmışdır. Verifikasiyanın dinamik metodlarından olan sınaq metodlarının müqayisəli təhlili aparılmışdır.**

Açar sözlər— verifikasiya metodları, sınaq metodlar, alfa sınağı, beta sınağı, ağ qutu sınağı, qara qutu sınağı

I. GİRİŞ

İnformasiya texnologiyaları (İT) müasir cəmiyyətin əsas elementlərindən biridir. İT iqtisadi fəaliyyət və cəmiyyətin sosial və mədəni inkişafı üçün bir baza olub, insanlara böyük həcmli müxtəlif informasiya massivləri ilə işləmək imkanı verir və onların harda olmalarından asılı olmayaraq bir-biriləri ilə əlaqə yaratmalarını təmin edir.

Son illərdə proqram təminatının (PT) işlənilmə texnologiyalarının sürətli inkişafı proqramçı əməyinin məhsuldarlığını, yəni onlar tərəfindən zaman vahidi ərzində yazılan proqram kodunun həcmini artırmışdır. PT-nin mürəkkəbliyinin artması onda olan səhvlərin sayının artmasına gətirir və sınaqdan keçirilməyən proqram kodunun min sətrinə düşən səhvlərin orta sayı 10-50 intervalında dəyişir [1]. PT-də olan ciddi səhvlər insan həyatının itirilməsinə, vaxt itkisinə, əlavə xərclərə və ya infrastruktur şəbəkələrinin işində böyük miqyaslı xətaların yaranmasına səbəb ola bilər. Bu tip səhvlərdən ilk olaraq 1962-ci ildə Mariner 1 kosmik aparatının idarə sistemində yaranan və onun itirilməsi ilə nəticələnən səhvi göstərmək olar. Məhz bu insidentdən sonra ABŞ hərbi-hava qüvvələrinin idarə heyəti PT-nin işlənilmə prosesində proqram kodunun ekspertizasının aparılmasını (proqramı işləyəndən sonra proqram kodunun başqa mütəxəssis tərəfindən təhlil edilməsi) qərara almışdır.

Müasir sistemlərin düzgünlüyünə və etibarlılığına təminat vermək üçün PT-nin həyat dövrünün müxtəlif mərhələlərində verifikasiya metodlarından istifadə edərək onda olan səhvlər ardıcıl şəkildə aradan qaldırılır. Verifikasiya standartların tələb etdiyi normalar, PT-yə qoyulan tələblərin siyahısı (texniki tapşırıq), layihə həlləri ilə ilkin kod, istifadəçi sənədləşməsi və PT-nin işləməsi arasındakı qarşılıqlı uyğunluğu yoxlayır. Bundan əlavə, tələblərin, layihə həllərinin, sənədlərin və proqram kodunun bu ölkədə qəbul edilmiş norma və standartlara uyğun tərtib edildiyi də yoxlanılır [2].

II. VERİFİKASİYA METODLARI

PT-nin verifikasiya metodları PT-nin xüsusiyyətlərinin qoyulan tələblərə uyğun gəldiyini təsdiq etmək üçün nəzərdə tutulmuşdur. Bu metodlar təyinatına və son nəticəyə çatma üsullarına görə müxtəlifdir. PT-nin verifikasiya metodlarını formal riyazi metodlara və ya PT-nin düzgün işləməsi barədə qərar qəbul edən şəxsin qiymətləndirməsindən asılı olan struktur və funksional metodlara bölmək olar. Proqram təminatının texniki vəziyyəti və iş qabiliyyətinin qiymətləndirilməsinə yönəlmiş verifikasiya metodları əsasən aşağıda göstərilən şəkildə təsnifatlandırılır [3]:

1. ekspertiza;
2. statik analiz;
3. formal metodlar;
4. dinamik analiz;
5. sintetik metodlar.

A. Ekspertiza

Ekspertiza (*review)* insanların bilik və təcrübəsinə, yəni ekspert qiymətləndirməsinə əsaslanır. PT-nin verifikasiyası zamanı ekspertizanı əsas fərqləndirən cəhət bu metod üçün beynəlxalq standartların mövcud olmasıdır (ISO 12207, IEEE 1074, ISO 15288, ISO 15504). Ekspertiza formal modellərə deyil PT-nin özünə tətbiq olunur.

Ekspertizanın aşağıdakı növləri var:

- ✓ təşkilati ekspertiza (*management review*);
- ✓ texniki ekspertiza (*technical review*);
- ✓ birbaşa nəzarət (*walkthrough*);
- ✓ inspeksiya (*inspection*);
- ✓ audit (*audit*).

Ekspertizanın müxtəlif formaları PT layihələndirilməsinin müxtəlif mərhələlərində tətbiq edilir. Ekspertiza metodları arasında arxitekturun sistematik analiz metodları daha geniş tətbiq edilir:

- proqram təminatı arxitekturunun analizi – SAAM (*Software Architecture Analysis Method*). Keyfiyyət göstəricilərini təhlil edir [4];
- arxitektur kompromislərinin analizi – ATAM (*Architecture Tradeoff Analysis Method*). ATAM SAAM-ın təkmilləşdirilmiş versiyasıdır. Keyfiyyət xarakteristikalarına aid tələblərin arxitektur həllərini analiz etməyə imkan verir [5, 6].
- konstruksiyanın aktiv analizi – ADR (*Active Design Review*). ADR yarımçıq qalmış və ya işlənilmə mərhələsində

DOI: 10.25045/NCSoftEng.2017.53

olan arxitektur üçün nəzərdə tutulur. Əsas fərqi ondadır ki, ümumi ananliz aparmır, ayrı-ayrı problemləri təhlil edir.

- aralıq konstruksiyaların aktiv analizi – ARID (*Active Reviews of Intermediate Designs*). ARID arxitekturu özündə ADR, ATAM və SAAM analizləri birləşdirir. [7].

Ekspertiza PT-nin həyat dövrünün istənilən prosesinə tətbiq edilə bilər, istənilən tip səhvləri ilkin mərhələlərdə aşkarlamağa imkan verir və bununla da PT-də yarana biləcək səhvləri minimuma endirir. Ekspertizanı avtomatlaşdırmaq olmur, burada insanların aktiv iştirakı vacibdir. Statistikaya görə PT-nin ilkin mərhələlərində aşkar edilən səhvlərin 50-90%-i ekspertizanın payına düşür [8].

B. Statik analiz

Statik analiz ilkin kodun düzgün yazılmasını və tez-tez rast gəlinən xətaları hər hansı bir şablon əsasında yoxlayır. Səhvlərin axtarılması əsasən avtomatlaşdırılmışdır və qərarların qəbul edilməsində insanın iştirakına ehtiyac qalmamaışdır. Buna baxmayaraq, statik analiz vasitəsilə aşkar edilən səhvlərin dairəsi məhduddur. Bu şablonlara misal olaraq proqramlaşdırma dilinin kompilyatoruna əlavə edilmiş semantik qaydaları göstərmək olar. Proqram kodunun statik analizinin köməyi ilə xətaların avtomatik axtarış sistemləri proqramların işlənilməsinin ilk mərhələlərində tətbiq edilir, bu halda səhvlərin düzəlişi ucuz başa gəlir.

C. Formal metodlar

Formal analiz metodları PT-nin riyazi modelləri və abstrakt təsvirləri ilə işləyir, proqramın fiziki cəhətdən icrasına tələb olmur, bu da rahat və səmərəlidir.

Bu metodları tətbiq etmək üçün PT-nin modeli düzgün qurulmalıdır və ancaq bu halda onun xüsusiyyətlərini düzgün təhlil etmək olur. Bəzi hallarda təhlilin səmərəliliyi üçün mütəxəssislərdən güclü riyazi məntiq və bu texnika ilə işləmə vərdişləri tələb olunur. Belə mütəxəssilərin sayı azdır və onların xidmətləri baha qiymətləndirilir. Formal modellərin qurulmasını avtomatlaşdırmaq olmur və insanın iştirakı həmişə labüddür. Buna baxmayaraq bəzi sahələrdə sistemdə olan səhvlər çox baha başa gəlir və ya fəlakətlə nəticələnir. Belə sistemlərin verifikasiyası üçün formal metodlardan istifadə edilir, çünki bu metod ekspertiza və sınaq vasitəsilə aşkar edilməyən səhvləri tapmağa qadirdir.

D. Dinamik analiz

Dinamik analiz metodları hazır məhsulu və ya onun prototipini analiz edir və PT-nin iş qabiliyyəti ilə əlaqədar yaranan problemləri üzə çıxarır. Dinamik metodları PT-nin layihələndirilməsi və işlənilməsi proseslərində tətbiq etmək olmaz. Bu metodlar ancaq proqram işləyərkən meydana gələn səhvləri aşkaramağa imkan verir. Dinamik metodlara sınaq və monitorinqi aid edirlər.

Verifikasiyanın dinamik metodlarını tətbiq etmək üçün testlərin hazırlanması və onların yerinə yetirilməsini təmin edən sınaq sisteminin işlənilməsi və ya yoxlanılan sistemin müəyyən xarakteristikalarına nəzarət etmək üçün monitorinq sisteminin yaradılması kimi işlər yerinə yetirilməlidir. Sınaq və monitorinq sistemləri bir dəfə işlənilir, lakin bir neçə dəfə tətbiq edilə bilər. Testlər isə hər yoxlanılan sistem üçün yenidən hazırlanır.

Monitorinq texnikaları PT-nin xarakteristikalarının qiymətləndirilmə üsuluna görə iki yerə bölünür:

- hadisələrə əsaslanan texnikalar (*event based*) (metrika kimi zaman, məzmun, istifadə olunan yaddaşın həcmi və s. götürülür);

- statistik texnikalar (*statistical*) sistemin işinin göstəricilərinə əsaslanır.

Monitorinq alətləri müxtəlifdir və hər biri adətən bir xarakteristikanı yoxlayır (PT-nin məhsuldarlığı, düzgünlüyü, təhlükəsizliyi və s.). Məsələn, Visual Studio Team System Profiler [9], Valgrind [10, 11], коммерческие Rational PurifyPlus [12] və Intel VTune Performance Analyzer [13].

E. Sintetik metodlar

Göstərilən metodlarla yanaşı yuxarıda adları çəkilən metodların kombinasiyasından ibarət olan sintetik metodlardan da istifadə edilir. Bu metodlara aşağıdakılar aiddir:

- ✓ modellərə əsaslanan sınaq;
- ✓ PT-nin formal xüsusiyyətlərinin monitorinqi. Bu sahədə iki termindən istifadə edilir (*runtime verification* və *passive testing*);
- ✓ formal xüsusiyyətlərin statik analizi metodu;
- ✓ struktur testlərinin generasiyasının sintetik metodları.

III. SINAQ METODLARI

Sınaq metodları test edilən sistemin və ya komponentin layihə həllərinə, tələblərə, layihənin işlənildiyi və müşayiət edildiyi ümumi məqsədlərə uyğunluğunu yoxlayır. Sınağın keçirildiyi hallar test vəziyyəti (*test situations, test purposes*), bu vəziyyətlərin yoxlanılmasını təsvir edən prosedurlara isə test deyilir.

PT-nin sınağı (*software testing*) – keyfiyyətə nəzarət texnikalarından biri olub proqramın gözlənilən və real işi arasındakı uyğunluğu müəyyən edir və seçilmiş testlər dəsti əsasında müəyyən qaydada həyata keçirilir. Burada test edilən sistemin və ya komponentin layihə həllərinə, tələblərə, layihənin işlənildiyi və müşayiət edildiyi ümumi məqsədlərə uyğunluğu yoxlanılır. Sınağın keçirildiyi hallar test vəziyyəti (*test situations, test purposes*), bu vəziyyətlərin yoxlanılmasını təsvir edən prosedurlara isə test deyilir. [14].

Sınaq verifikasiyanın dinamik analiz metodlarına aiddir, o PT-nin bütün həyat dövründə müxtəlif səviyyələrdə aparılır. Sınağın səviyyəsi onun ayrı-ayrı komponentlərə, bir qrup birləşmiş komponentlərə və ya bütövlükdə sistemə tətbiq ediləciyini müəyyən edir.

A. Yoxlanılan elementlərin səviyyəsinə və miqyasına görə sınaq metodları

Yoxlanılan elementlərin səviyyəsinə və miqyasına görə sınaq metodları aşağıdakı növlərə bölünür:

- ***modul və ya komponent sınağı*** (*unit testing, component testing*). Ən aşağı sınaq səviyyəsidir. Proqram təminatının test oluna bilən ən kiçik komponentlərini (*unity*) yoxlayır. Yoxlanılan bir metod və ya bir funksiya ola bilər.

DOI: 10.25045/NCSoftEng.2017.53

Bu komponentlər daha sonra sistemdə birləşdirilərək inteqrasiya testi vastəsilə sınaqdan keçirilir. İdeal halda komponent testləri bir-birindən asılı olmurlar;

- ***inteqrasiya sınağı*** (*integration testing*). İnteqrasiya sınağı mərhələsində komponentlər birləşdirilir və inteqrasiya olunmuş komponentlər qrup şəklində test edilirlər. Ayrılıqda inteqrasiya mərhələsində səhvsiz işləyən komponentlərin qarşılıqlı əlaqəsi zamanı səhvlər yarana bilər;

- ***sistem sınağı*** (*system testing*). İnteqrasiya olunmuş komponentlərdən ibarət sistemin PT-nin spesifikasiyalarında göstərilən tələblərə cavab verməsi yoxlanılır. Test mühiti ilə real istehsal mühiti mümkün olduğu qədər eyni olmalıdır və ya ona uyğun hazırlanmalıdır;

- ***qəbul edilmə sınağı*** (*acceptance testing*). PT istifadəyə verildikdən sonra real mühitdə sınaqlar aparıldıqdan sonra sifarişçi tərəfindən qəbul edilə bilər [15].

B. Məqsədinə görə sınaq metodları

Məqsədindən asılı olaraq PT-nin sınaq metodları şərti olaraq aşağıdakı növlərə bölünür:

1. funksional sınaq. Funksiyaları və xüsusiyyətləri, eyni zamanda digər sistemlərlə qarşılıqlı əlaqədə işləmə qabiliyyətini yoxlayır və sınağın yuxarıda göstərilən bütün növlərində tətbiq edilə bilər. Funksional sınaq növlərindən ən geniş yayılanlar aşağıda göstərilmişdir:

- funksional sınaq (*functional testing*);
- təhlükəsizlik sınağı (*security and access control testing*);
- qarşılıqlı işləmə sınağı (*interoperability testing*).

2. qeyri-funksional sınaq. Qeyri-funkional sınaq PT-nin müxtəlif qiymətlərlə ölçülə bilən xarakteristikalarını yoxlayan testlərdən ibarətdir.

Qeyri-funksional sınaq növlərindən ən geniş yayılanlar aşağıda göstərilənlərdir:

- məhsuldarlığı yoxlayan bütün sınaqlar:
  - ✓ perfomans sınağı (*performance and load testing*)
  - ✓ stress sınağı *(stress testing)*
  - ✓ dayanıqlılıq və etibarlılığın sınağı (*stability / reliability testing*)
  - ✓ kodun həcminin sınağı (*volume testing*)
- PT-nin qurulmasının sınağı (*installation testing*);
- istifadənin rahatlığının sınağı (*usability testing*);
- imtina və bərpa olunmanın sınağı (*failover and recovery testing*);
- konfiqurasiyanın sınağı (*configuration testing*).

3. dəyişikliklərlə əlaqədar sınaq. Proqram üzərində lazımi dəyişikliklər (səhvlərin düzəldilməsi) problemin aradan qaldırıldığından əmin olmaq üçün PT yenidən sınaqdan keçirilməlidir. Aşağıda PT-nin qurulmasından sonra onun iş qabiliyyətini və edilən düzəlişlərin düzgün yerinə yetirildiyini yoxlamaq üçün sınaq növləri göstərilmişdir:

- tüstü sınağı (*smoke testing*);
- reqressiya sınağı (*regression testing*);
- qurulmuş PT-nin sınağı (*build verification test*);
- səhvsiz işləmə və ya uyğunluğun sınağı (*sanity testing*).

C. Sınaq meyarları

PT-nin sınağı zamanı elə meyarlar seçilməlidir ki, sınağın sadə olmasını və testlərin məqsədyönlü seçilməsini təmin etsin. Strukturun sınağı zamanı PT-nin tamlığının ən zəif meyarı proqramın hər operatorunun heç olmasa bir dəfə icra edilməsi tələbidir. Daha güclü meyar alqoritmin hər qolunun (bütün keçidlər) işinin heç olmasa bir dəfə yoxlanılması tələbidir. Lakin bu sınaq tələblərinə diqqətlə fikir verdikdə onların bir-birini tamamladığı görünür, hər metod müxtəlif növ səhvlərin tapılmasına xidmət edir. Ona görə də təcrübədə "ağ qutu" və "qara qutu" sınaqlarından və ya onların kombinasiyası olan "boz qutu" sınağından istifadə edilir.

Testlərin yaradılması üçün götürülən meyarlara və informasiya mənbələrinə görə klassifikasiya etdikdə sınaq metodlarını aşağıdakı növlərə bölmək olar:

1. ağ qutu sınağı (*white-box testing, glass-box testing* və ya *structural testing*). Proqramı yoxlayan proqramın daxili strukturunu və ilkin kodunu oxumağı bilməlidir, çünki bu metod proqramın daxili strukturunu yoxlayır. Bu sınaq metodunda proqram kodunda olan hər bir proqram yolu *(programm path)* ən azı bir dəfə yerinə yetirilməlidir və hər moduldan və ya hər funksiyadan ən azı bir dəfə istifadə olunmalıdır.

Ağ-qutu sınağında əminliklə demək olur ki, proqramda sınaqdan keçirilməmiş qol yoxdur və proqramı sınaqdan keçirən bu sahə üzrə peşəkar mütəxəssis olduğundan proqramın hansı hissəsinin daha diqqətli şəkildə yoxlanılacağını bilir. Lakin bu metoddakı prosedurlar qara qutu sınağında olduğundan daha çox xərc tələb edir və ancaq kodda olan səhvləri yoxladığından proqramın qoyulan spesifikasiyalara uyğunluğunu təyin edə bilmir.

2. ***qara qutu sınağı*** (*black-box testing*), buna bəzən uyğunluq testi (*conformance testing*) və ya funksional test də (*functional testing*) deyilir. Qara qutu sınağında proqramı yoxlayan hansı funksiyanın yerinə yetiriləcəyini bilir, o funksiyanın necə işləməsindən və proqram kodundan xəbərdar olmur. Sınağı aparan proqramlaşdırma biliklərinə sahib olmalı deyil. O yalnız PT-nin istifadəçi təlimatına və proqram spesifikasiyalarına uyğunluğunu yoxlamalıdır.

Qara-qutu sınağında PT-ni sınaqdan keçirən instrumental vasitələri və ilkin kodu bilmək məcburiyyətində deyil və bu metodda yanaşma üsulu ağ qutu sınağındakından daha sadədir. Lakin proqramın lazımsız və kritik hissələrinin olub-olmadığını təyin etmək mümkün deyil.

3. ***boz qutu sınağı*** (*grey-box testing*). Səhvlərin aradan qaldırılmasına xidmət edən effektiv test hallarının yaradılması üçün qara və ağ qutu sınağının kombinasiyası məsləhət görülür, çünki hər metodun öz çatışmazlıqları və üstünlükləri

DOI: 10.25045/NCSoftEng.2017.53

vardır. Bu kombinasiyanı *gray-box test* və ya *broken-box* adlandırırlar [15].

D. Sınaq mərhələləri

Alfa və beta sınaqları proqram sənayesində geniş yayılmış terminlərdəndir və sınaq təcrübəsində hər ikisinin öz fəaliyyət sahəsi və özəllikləri vardır. Microsoft və İBM kimi şirkətlər öz proqram məhsullarını bazara çıxararkən onların məhsulları son istifadəçiyə çatmazdan öncə alfa və beta sınağından keçirlər:

1. ***Alfa sınağı*** mərhələsində yoxlama peşəkar mütəxəssis qrupları tərəfindən aparılır. O, aşağıda göstərilən xüsusiyyətlərə malikdir:

- peşəkar mütəxəssis və proqramı işləyənlər tərəfindən gerçəkləşdirilir;
- bir-birindən aslı olmayan sınaq qrupları tərəfindən həyata keçirilir;
- bu test bazara və ictimaiyyətə açıq deyildir;
- tətbiqi proqramlar və layihələr üçün həyata keçirilir;
- həmişə virtual mühitdə yerinə yetirilir;
- proqramı işləyən təşkilat çərçivəsində həyata keçirilir;
- həm ağ qutu həm də, qara qutu sınağı kateqoriyasındandır;
- PT-nin qəbul edilməsinin sınağı zamanı (proqramı işləyənlərin məhsulun və layihənin istifadəçinin tələbinə uyğun gəlib gəlmədiyini yoxlaması üçün) yerinə yetirilir;
- istifadəçinin olmadığı auditoriyada həyata keçirilir [16,17].

Alfa sınağın yerinə yetirilməsi erkən mərhələdə proqramın etibarlılığı haqqında fikir yaradır, dizayn və funksionallıqla bağlı səhvləri aşkar etməyə imkan verir və real vaxt mühitini simulyasiya edir. Lakin proqram hələ inkişaf mərhələsində olduğundan PT-nin funksionallığı tam olaraq sınaqdan keçirilə bilmir və bəzi hallarda istifadəçilər alfa testin nəticələrindən narazı qalırlar [18].

2. Beta sınağı zamanı alfa sınaqdan keçmiş proqram məhsullarının necə işlədiyini yoxlamaq üçün onu safarişçi və ya son istifadəçinin ixtiyarına verirlər. Beta sınaq aşağıda göstərilən xüsusiyyətlərə malikdir:

- müştərilər tərəfindən onların öz məkanlarında gerçəkləşdirilir;
- proqramı işləyən təşkilatdan asılı olmayan sınaq qrupları tərəfindən həyata keçirilir;
- bazara və ictimaiyyətə açıqdır;
- proqram məhsulu üçün həyata keçirilir;
- real vaxt mühitində yerinə yetirilir;
- proqramı işləyən təşkilat xaricində həyata keçirilir;
- yalnız qara qutu sınağı kateqoriyasındandır;
- məhsul və layihə bazara çıxarıldığı zaman yerinə yetirilir;
- proqramı işləyənlərin olmadığı istifadəçi auditoriyasında gerçəkləşdirilir [18].

Beta testi həmçinin, sahə testi (*field testing*) adı ilə də tanınır.

Beta testin müştəri tərəfindən qiymətləndirməsi məhsulda olan uğursuzluq riskini azaldır. Eyni zamanda müştərinin öz fikirlərini bildirməsi məhsulun keyfiyyətini yaxşılaşdırmağa imkan verir. Lakin şirkət daxilində PT-ni işləyənlərin nəzarəti altında həyata keçirilən digər test növlərindən fərqli olaraq beta testi real mühitdə həyata keçirilir və nadir hallarda ona nəzarət etmə imkanı olur [19, 20].

NƏTICƏ

Sınaq metodlarının analizi nəticəsində məlum oldu ki, alfa və beta sınaqlardan birgə istifadə daha məqsədəuyğundur.

Verifikasiyanın əsas məqsədləri aşağıdakı kimi müəyyən edilmişdir:

- PT-nin həyat dövrünün müxtəlif mərhələlərində xətaların (səhvlər, proqramın natamam işlənilməsi, tələblərin düzgün təsvir edilməməsi və s.) aşkar edilməsi;
- yaradılan və ya müşayiət edilən proqram sisteminin daha kritik və səhvlərə daha çox meylli olan hissələrinin tapılması;
- PT-nin işində marağı olan bütün şəxslərə (rəhbərlər, sifarişçilər, istifadəçilər və s.) layihənin cari vəziyyəti və onun nəticələrinin xarakteristikaları haqqında məlumat verilməsi;
- layihə rəhbərlərinə və proqram mühəndislərinə növbəti işlərin planlaşdırılması üçün, eyni zamanda layihənin davam etdirilməsi, dayandırılması və ya sifarişçiyə təhvil verilməsi barədə qərarın qəbul edilməsi üçün informasiyanın verilməsi.

Proqram təminatında verifikasiya və sınaq metodlarından istifadə edilməsi proqramda olan səhvlərin vaxtında aşkarlanmasına və onun keyfiyyətinin yüksəldilməsinə səbəb olur. Buna baxmayaraq elə bir standart metod yoxdur ki, onun vasitəsi ilə bütün səhvlər aradan qaldırılsın.

PT-nin keyfiyyətinin yüksəldilməsi üçün yüksək biliyə və təcrübəyə sahib olan mütəxəssislərə ehtiyac duyulur. Azərbaycan ali təhsil müəssisələrində proqram mühəndisliyi fənni daxilində verifikasiya və sınaq metodlarının da tədris edilməsi arzu ediləndir.

ƏDƏBİYYAT

[1] The Economic Impacts of Inadequate Infrastructure for Software Testing. NIST Report, p. 309, 2002

[2] IEEE 1012-2004 Standard for Software Verification and Validation. IEEE, 2005

[3] В. В. Кулямин. Методы верификации программного обеспечения. Всероссийский конкурс обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 111 с, 2008

[4] B. Boehm, V. Basili. Software Defect Reduction Top 10 List. IEEE Computer, vol. 1, pp.:135-137, January 2001

[5] D. L. Detlefs, K. R. M. Leino, G. Nelson, J. B. Saxe. Extended static checking.Technical Report SRC-RR-159, Digital Equipment Corporation, Systems Research Center, 1998

[6] «Анализ архитектуры». http://brtrg.by/blog/post/152

[7] "Active Reviews for Intermediate Design", http://www.sei.cmu.edu/architecture/tools/evaluate/arid.cfm

DOI: 10.25045/NCSoftEng.2017.53

[8] O. Laitenberger. A Survey of Software Inspection Technologies. In Handbook on Software Engineering and Knowledge Engineering, v. 2, pp. 517-555. World Scientific Publishing, 2002

[9] http://msdn.microsoft.com/en-us/library/z9z62c29(VS.80).aspx

[10] http://valgrind.org/

[11] N. Nethercote, J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. Proc. of 2007 PLDI conference, ACM SIGPLAN Notices, v.6, No42, pp.89- 100, June 2007

[12] http://www-306.ibm.com/software/awdtools/purifyplus/

[13] http://www.intel.com/cd/software/products/asmo-na/eng/239144.htm.

[14] D. R. Cok, J. R. Kiniry, ESC/Java2: Uniting ESC/Java and JML. Proc. of International Workshop on the Construction and Analysis of Safe, Secure, and Interoperable Smart Devices (CASSIS'04), LNCS 3362: pp.108-128, Springer-Verlag, January 2005

[15] “SWEBOKv3” http://www4.ncsu.edu/~tjmenzie/cs510/pdf/SWEBOKv3.pdf

[16] “Softwaretests”, https://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2010_2011/siw-1011-ehmke-tests-ausarbeitung.pdf

[17] “Verifikation, Validation und Testen von Sicherheitskritischen Systeme”, http://pi.informatik.uni-siegen.de/niere/lehre/SS04/ SeminarFinal/ 5_jaya/Ausarbeitung.pdf

[18] Alpha Testing Vs Beta Testing”, http://www.guru99.com/alpha-beta-testing-demystified.html

[19] “Best Difference between Alpha and Beta Testing”, http://testingbasicinterviewquestions.blogspot.com/2015/03/best-difference-between-alpha-and-beta.html

[20] С. Макконнелл, Совершенный код. М.: Русская редакция, 2005, 896 с.

DOI: 10.25045/NCSoftEng.2017.53