چهارمین کنفرانس ملی
فناوری اطلاعات، کامپیوتر و مخابرات
4th National Conference On Information Technology,Computer & Telecommunication
July 13 2017
۲۲ تیر ۱۳۹۶

بررسی تست جعبه سیاه و طبقه بندی نقص شناسایی شده در آن

محمد لطافت[۱]

[۱] **دانشجوی کارشناسی ارشد، دانشکده‌ی برق و کامپیوتر، دانشگاه صنعتی قم**

Letafat.m@qut.ac.ir

محبوبه شمسی[۲]

[۲] **استادیار دانشکده‌ی برق و کامپیوتر، دانشگاه صنعتی قم.**

Shamsi@qut.ac.ir

عبدالرضا رسولی کناری[۳]

[۳] **استادیار دانشکده‌ی برق و کامپیوتر، دانشگاه صنعتی قم.**

Rasouli@qut.ac.ir

چکیده

با توجه به رشد روز افزون توسعه نرم افزار و نیازهای اساسی جامعه به محصولات نرم افزاری ، فرآیند تست نرم افزار هم از لحاظ کیفی و هم از لحاظ قابلیت اطمینان بسیار حائز اهمیت است. نقص نرم افزار معمولا با بررسی تست جعبه سیاه و جعبه سفید شناسایی می شود. کار شناسایی نقص نرم افزار و استخراج الگوهای نقص و طبقه بندی انواع مختلف نقص ما را در توسعه نرم افزار هدایت می کند. زیرا با طبقه بندی نقص ها تمرکز بیشتری روی آنها خواهیم داشت. در این مقاله در ابتدا ابزار تولید موارد آزمون مختلف به صورت خودکار بیان شده سپس طرح طبقه بندی نقص ODC-BD معرفی و الگوی نقص شناسایی می شود.

کلمات کلیدی

تست نرم افزار، تست جعبه سیاه، طبقه‌بندی نقص، تحلیل نقص، نقص جعبه سیاه، استخراج الگو، قابلیت اطمینان.

مقدمه

در فرآیند توسعه نرمافزار، تست نرمافزار با هدف افزایش قابلیت اطمینان ذینفعان نرمافزار در مورد کیفیت محصول یا خدمات انجام می شود. منظور از قابلیت اطمینان نرمافزار احتمال عملکرد بدون شکست سیستم می باشد. قابلیت اطمینان به جنبه های مختلفی از جمله تست نرمافزار وابسته است.

به طور کلی سه دسته تست نرمافزار وجود دارد:

- تست جعبه سفید
- تست جعبه خاکستری
- تست جعبه سیاه

تمرکز اصلی ما در این مقاله، بر تست جعبه سیاه است. آن دست از معایبی که توسط این تست شناسایی می‌شوند، نقص جعبه سیاه نامیده می‌شود. به طورکلی نقص نرمافزار عدم تطابق بین نتیجه مورد انتظار و نتیجه واقعی نرمافزار می باشد. تست جعبه سیاه اشاره به روش آزمایش یک سیستم بدون دانش از داخل آن سیستم می باشد. این تست دسترسی به کد برنامه، معماری برنامه و نحوه قرارگیری اجزای برنامه ندارد و فقط از طریق یک رابط کاربری و ارتباط بین ورودی و خروجی بدون دانستن این‌که این ورودی چگونه به خروجی تبدیل شده است، به آزمون نرم‌افزار می‌پردازد. در تست جعبه سیاه هدف اعمال ورودی و بررسی صحت خروجی می باشد که عدم صحت خروجی، نقص نرمافزار تلقی می شود. برای ایجاد اطمینان از قابلیت عملکردی سیستم تست جعبه سیاه باید مطابق با مشخصات عملیاتی و کارکردی سیستم انجام شود[۱]. در ابتدا به فرآیند توسعه نرمافزار و تولید خودکار موارد آزمون و نتایج یک آزمون تجربی می پردازیم در گام بعد به معرفی طرح طبقه بندی ODC-BD و در نهایت آشنایی به چهار مرحله شناسایی و طبقه بندی نقص توسط ODC-BD می پردازیم.

فرآیند توسعه نرم افزار

توسعه نرم فزار در صنعت نرمافزار به طور اساسی شامل چهار مرحله، بررسی نیازمندی ها، طراحی، پیاده سازی و تست و نگهداری است که روند آن را در شکل۱ مشاهده می کنید.

شکل ۱مراحل توسعه نرم افزار(K. Kaur and S. Sharma,2015)

بررسی نیازمندی ها

مهندسی نیازمندی‌ها، فرآیندی است که به شناسایی نیازهای مشتریان می‌پردازد؛ به طوری که نرم‌افزار با مدنظر قرار دادن نیازهای مشخص شده و با هدف برطرف کردن آن‌ها، طراحی شده و توسعه می‌یابد.

طراحی

زمانی که شناسایی نیازها، به طور کامل انجام شد، وارد مرحله‌ی طراحی می‌شویم. طراحی، یک الگو با قابلیت استفاده‌ی مجدد است که راه حل و چگونگی برخورد با نیازهای شناسایی شده، در این مرحله مشخص می‌گردد.

پیاده سازی

در این مرحله توسعه واقعی سیستم صورت می گیرد. کل پروژه به ماژول هایی شکسته می شود و با توجه به تخصص تیم های برنامه نویسی پیاده سازی هر ماژول به یکی از تیم ها سپرده می شود.

تست نرم افزار

پیدا کردن اشتباهات و خطاهای مرحله طراحی و برنامه نویسی در این مرحله صورت می گیرد، در طول مرحله تست ما به دنبال یافتن خطاها و اشکالات موجود در نرمافزار و برطرف کردن آنها می باشیم. تست نرمافزار به صورت زیر تعریف می شود: فرآیندی که در آن بررسی می شود که آیا نرمافزار تولید شده نیازهای شناسایی شده در مرحله اول را برطرف می کند یا خیر؟ برای این منظور سه نوع تست وجود دارد:

- تست جعبه سفید
- تست جعبه خاکستری
- تست جعبه سیاه

در تست جعبه سفید به بررسی دقیق ساختار داخلی برنامه و کد برنامه توجه می شود. برای انجام این تست نیاز به دانش کامل از منطق برنامه می باشد.

در تست جعبه سیاه عملکرد سیستم مورد ارزیابی قرار می گیرد. این تست برای یافتن خطاها در ساختار داده، توابع معیوب، خطاهای رابط کاربری و غیره می باشد. این تست مکانیزم داخلی سیستم را نادیده گرفته و تنها اشکالات موجود در عملکرد سیستم را با توجه به خروجی نادرست شناسایی و نمایش می دهد.

منظور از تست جعبه خاکستری که ترکیبی از دو تست جعبه سفید و سیاه می باشد، یعنی در این روش تست به میزان محدود باید از منطق و ساختار داخلی برنامه اطلاع داشت و هم باید عملکرد سیستم در برابر ورودی بررسی کرد. به طور کلی در جدول شماره۱ مقایسه این سه نوع تست ارائه شده است.

جدول ۱مقایسه سه روش تست)M. E. Khan and F. Khan,2012, D. S. Singh, T. Hussain 2015(

ردیف	تست جعبه سیاه	تست جعبه خاکستری	تست جعبه سفید
۱	تحلیل عملکرد بدون دانش از نحوه کارکرد	دانش جزئی از نحوه کارکرد	دانش کامل از نحوه عملکرد
۲	تعداد ماژول کم	تعداد ماژول متوسط	تعداد ماژول زیاد
۳	توسط کاربران و تست کننده برنامه و توسعه دهندگان انجام می شود(تست پذیرش کاربر)	توسط کاربران و تست کننده برنامه و توسعه دهندگان انجام می شود(تست پذیرش کاربر)	توسط توسعه دهندگان و تست کننده برنامه انجام می شود.
۴	تست بر اساس رفتار خارجی سیستم بدون درنظر گرفتن روابط داخلی انجام می شود.	طراحی موارد آزمون بر پایه الگوریتم های سطح بالا، آگاهی مبتنی بر الگوریتم و معماری برنامه	دانش کامل از روابط داخلی
۵	جامع و و وقت‌گیر	جامع و وقت‌گیر در حد متوسط بین دو تست	به طور بالقوه جامع و وقت-گیر
۶	برای تست الگوریتم مناسب نمی باشد.	برای تست الگوریتم مناسب نیست.	برای تست همه الگوریتم ها مناسب است.

در این مقاله تمرکز برروی تست جعبه سیاه می باشد، که برای آشنایی با برخی نکات این تست به جدول۲ مراجعه نمائید.

جدول ۲ نکات تست جعبه سیاه(L. Williams,2006)

ردیف	توضیحات
۱	این تست بدون توجه به ساختار داخلی برنامه و با توجه به نیاز و خواسته های کاربر انجام می شود. بهتر است که این تست توسط کسی به غیر از توسعه دهنده برنامه انجام شود، تا عملکرد واقعی سیستم بررسی شود.
۲	موارد آزمون بر اساس چهار مورد(شماره، توضیحات، خروجی مورد انتظار کاربر و خروجی واقعی) برنامه ریزی شده باشند.
۳	این تست فقط وجود یا عدم وجود خطا را نمایش می دهد. کشف محل وقوع نقص برعهده توسعه دهنده نرم‌افزار است.
۴	این تست در مراحل ابتدایی توسعه نرم‌افزار کاربرد دارد زیرا کشف و اصلاح خطا قابل انجام است، اما کشف خطا در مراحل پایانی توسعه نرم‌افزار بسیار دشوار است.
۵	استفاده از کلاس هم ارزی برای بخش بندی و مدیریت ساده تر موارد آزمون .
۶	تجزیه و تحلیل مقادیر مرزی برای یافتن اشکالات

A. تجزیه و تحلیل نقاط مرزی(BVA[1])
B. تست پایداری
C. تقسیم بندی به گروه های هم ارز
D. نمودارهای علت و معلول
E. تست مبتنی بر مدل

در این مقاله ما به بررسی تست جعبه سیاه و بررسی عملکرد دو روش BVA و تست پایداری می پردازیم.
به منظور نمایش کارایی دو روش BVA و تست پایداری یک ابزار توسعه یافته در C++ را برای تولید خودکار موارد آزمون مورد استفاده قرار گرفته است و برای بررسی اعتبار از مسئله خط راست استفاده کرده ایم. معادله خط y=mx+c که در آن برای , (m1,c1) (m2,c2) شرایط زیر برقرار است [۶]:

دو خط موازی هستند اگر c1≠c2, m1=m2
دو خط متقاطع هستند اگر m1 ≠ m2
دو خط منطبق هستند اگرm1=m2, c1=c2

حال به بررسی کارایی دو روش BVA و تست پایداری می پردازیم.
روش BVA بر پایه مقادیر روی لبه ورودی می باشد و اعتقاد دارد که بسیاری از خطاها روی نقاط مرزی و در وسط دامنه رخ می دهد به همین دلیل تمرکز اصلی این روش برروی نقاط مرزی می باشد. برای مثال محدود اعداد [10,100] مقادیر تست ۱۰، ۱۱، ۹۹، ۱۰۰ می باشد.
برای بررسی این روش دو خط y1و y2 را در نظر داریم که به ترتیب بین A,B و C,D می باشند. برای بررسی روش BVA ما موارد آزمون را با استفاده از ابزاری ایجاد کرده ایم و کل موارد آزمون و خروجی مورد انتظار در جدول۳ بیان شده است. به طور کلی در این روش موارد آزمون 4n+1 می باشد که در اینجا n=4 می باشد که در کل۱۷ مورد آزمون داریم.

جدول 1 موارد آزمون برای روش تحلیل نقاط مرزی(M. A. Khan and M. Sadiq,2011)

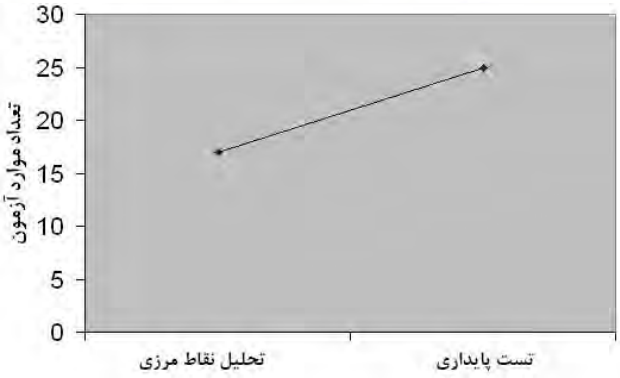
موارد آزمون	M1	C1	M2	C2	خروجی مورد انتظار
۱	۰	۵۰	۵۰	۵۰	دو خط متقاطع اند
۲	۱	۵۰	۵۰	۵۰	دو خط متقاطع اند
۳	۵۰	۵۰	۵۰	۵۰	دو خط منطبق اند
۴	۹۹	۵۰	۵۰	۵۰	دو خط متقاطع اند
۵	۱۰۰	۵۰	۵۰	۵۰	دو خط متقاطع اند
۶	۵۰	۰	۵۰	۵۰	دو خط موازی اند
۷	۵۰	۱	۵۰	۵۰	دو خط موازی اند
۸	۵۰	۹۹	۵۰	۵۰	دو خط موازی ان
۹	۵۰	۱۰۰	۵۰	۵۰	دو خط موازی ان
۱۰	۵۰	۵۰	۰	۵۰	دو خط متقاطع اند
۱۱	۵۰	۵۰	۱	۵۰	دو خط متقاطع اند
۱۲	۵۰	۵۰	۹۹	۵۰	دو خط متقاطع اند
۱۳	۵۰	۵۰	۱۰۰	۵۰	دو خط متقاطع اند
۱۴	۵۰	۵۰	۵۰	۰	دو خط موازی ان
۱۵	۵۰	۵۰	۵۰	۱	دو خط موازی ان
۱۶	۵۰	۵۰	۵۰	۹۹	دو خط موازی ان
۱۷	۵۰	۵۰	۵۰	۱۰۰	دو خط موازی ان

تست پایداری بر این موضوع اعتقاد دارد که یک ماژول زمانی پایدار و قابل اعتماد می باشد که در برابر هر ورودی شکست نخورد چون شکست یک ماژول برابر با شکست کل نرمافزار می باشد. خلاصه ایی از موارد آزمون و خروجی مورد انتظار در جدول۴ آورده شده است.

جدول 2 موارد آزمون برای روش تست پایداری(M. A. Khan and M. Sadiq,2011)

موارد آزمون	M1	C1	M2	C2	خروجی مورد انتظار
۱	-۱	۵۰	۵۰	۵۰	دو خط متقاطع اند
۲	۰	۵۰	۵۰	۵۰	دو خط متقاطع اند
۳	۱	۵۰	۵۰	۵۰	دو خط متقاطع اند
۴	۵۰	۵۰	۵۰	۵۰	دو خط منطبق اند
۵	۹۹	۵۰	۵۰	۵۰	دو خط متقاطع اند
۶	۱۰۰	۵۰	۵۰	۵۰	دو خط متقاطع اند
۷	۱۰۱	۵۰	۵۰	۵۰	دو خط متقاطع اند
۸	۵۰	-۱	۵۰	۵۰	دو خط موازی اند
۹	۵۰	۰	۵۰	۵۰	دو خط موازی اند
۱۰	۵۰	۱	۵۰	۵۰	دو خط موازی اند
۱۱	۵۰	۹۹	۵۰	۵۰	دو خط موازی اند
۱۲	۵۰	۱۰۰	۵۰	۵۰	دو خط موازی اند
۱۳	۵۰	۱۰۱	۵۰	۵۰	دو خط موازی اند
۱۴	۵۰	۵۰	-۱	۵۰	دو خط متقاطع اند
۱۵	۵۰	۵۰	۰	۵۰	دو خط متقاطع اند
۱۶	۵۰	۵۰	۱	۵۰	دو خط متقاطع اند
۱۷	۵۰	۵۰	۹۹	۵۰	دو خط متقاطع اند
۱۸	۵۰	۵۰	۱۰۰	۵۰	دو خط متقاطع اند
۱۹	۵۰	۵۰	۱۰۱	۵۰	دو خط متقاطع اند
۲۰	۵۰	۵۰	۵۰	-۱	دو خط موازی اند
۲۱	۵۰	۵۰	۵۰	۰	دو خط موازی اند
۲۲	۵۰	۵۰	۵۰	۱	دو خط موازی اند
۲۳	۵۰	۵۰	۵۰	۹۹	دو خط موازی اند
۲۴	۵۰	۵۰	۵۰	۱۰	دو خط موازی اند
۲۵	۵۰	۵۰	۵۰	۱۰۱	دو خط موازی اند

به طور کلی در شکل۲ عملکرد دو روش در تولید موارد آزمون نشان داده شده است.



شکل ۲مقایسه کارایی دو روش(M. A. Khan and M. Sadiq,2011)

همانگونه که در شکل۳ دیده می شود تولید موارد آزمون در تست پایداری بیشتر از تولید موارد آزمون با استفاده از تجزیه و تحلیل نقاط مرزی می باشد. جدول شماره۵ مقایسه این دو روش را نمایش می دهد

جدول 3 **نتایج آزمون معادله خط با استفاده از دو روش تست پایداری و تحلیل نقاط مرزی**)M. A. Khan and M. Sadiq,2011(

شماره	نوع تست	تعداد موارد آزمون
۱	تحلیل مقادیر مرزی	۱۷
۲	تست پایداری	۲۵
نوع خط	تحلیل مقادیر مرزی	تست پایداری
خط متقاطع	۸	۱۲
خط منطبق	۱	۱
خط موازی	۸	۱۲

طبقه بندی نقص

حال با توجه به اهمیت تست نرم‌افزار، شناسایی و رفع اشکالات و خطاها در آن، از اهمیت به‌سزایی برخوردار است. بدیهی است اگر این نقص‌ها، دسته‌بندی شده باشند، توسعه‌ی نرم‌افزار آسان‌تر می‌گردد؛ زیرا با طبقه‌بندی نقایص، شناسایی آن‌ها ساده‌تر و تمرکز بر روی آن‌ها بیش‌تر می‌شود.

روش طبقه بندی نقص با توجه به اهداف مختلف متفاوت می باشد، روش طبقه بندی ODC-BD[1] که توسعه یافته روش طبقه بندی ODC[2] می باشد، که با اهداف زیر توسعه داده شده است:

۱. تجزیه و تحلیل نقص های جعبه سیاه توسط مدیران تست
۲. برای بهبود بهره وری تست های جعبه سیاه

روش طبقه بندی ODC-BD توسعه یافته روش ODC می باشد در جدول شماره۶ تمام ویژگی های این دو روش بیان شده است.

جدول ۴ مقایسه دو روش طبقه بندی(N. Li, Z. Li, and L. Zhang,2010)

ویژگی	توضیحات برای ODC-BD	توضیحات برای ODC
تابع ماژول	نشان دهنده ارتباط ماژول	چنین ویژگی ندارد
نوع نقص	نشان دهنده نوع نقص داده و یا عملکرد	نشان دهنده علل نقص
عمر	نشان دهنده تاریخ تست در زمان نقص	نشان دهنده تاریخ شناسایی نقص
کیفیت	نشان دهنده کیفیت نرم افزار تحت تاثیر نقص	نشان دهنده میزان تاثیر نقص بر روی مشتری
ویژگی	نشان دهنده اینکه آیا نقص منجر به خطا می شود یا خیر	چنین ویژگی ندارد
اولویت پردازش	نشان دهنده اولویت گزارش دادن نقص در همان ابتدای وقوع نقص	چنین ویژگی ندارد
میزان نفوذ	نشان دهنده میزان تاثیر نقص بر روی کاربر	چنین ویژگی ندارد

به طور کلی روش ODC، برای توسعه‌دهندگان و روش ODC-BD برای کسانی که تست نرم‌افزار را انجام می‌دهند، طراحی و ارائه شده‌اند.

به طور کلی طبقه بندی نقص جعبه سیاه به صورت تجربی شامل چهار مرحله زیر می باشد(N. Li, Z. Li, and a. X. Sun,2010):

۱. جمع آوری داده های دارای نقص
۲. طبقه بندی نقص ها با استفاده از متدODC-BD
۳. دسته بندی نتایج طبقه بندی
۴. کالیبره کردن و بررسی مجدد طبقه بندی

در مرحله اول تمام گزارشات نقص جعبه سیاه موجود در پروژه های مورد نظر جمع آوری می شود.
در مرحله دوم گزارشات نقص مرحله قبل و یک خلاصه مختصر از نقص ها را برای متدODC-BD ارسال می کنیم، طبقه بندی این نقص به شرح زیر می باشد:

مرحله۱: ایجاد یک دسته از نوع نقص که اساس دسته بندی نقص ها می باشد.
مرحله۲: طبقه بندی نقص انتخاب شده با توجه به دسته های موجود، که در نهایت به دسته مناسب خود اضافه می شود. شرح نقص معمولا شامل داده ها و عملکرد و نتیجه نقص می باشد. حال با توجه نقص انتخابی تشخیص می دهیم که نقص مورد نظر در طبقه بندی ها جزوء دسته ایی می باشد و یا یک نقص خاص می باشد که خود تشکیل یک دسته را می دهد.
مرحله۳: نقص های طبقه بندی شده و دسته بندی شده را توسط موارد آزمون مجدد تست کرده و طبقه بندی نهایی بر اساس نوع نقص جعبه سیاه را انجام می دهیم.

مرحله سوم با استفاده از مرحله طبقه بندی و جزئیات طبقه بندی که مهم ترین آن نوع نقص است چهار سطح نقص تشکیل می دهیم که شامل: نقص تابع، نقص سیستم، نقص داده و نقص رابط کاربری می باشد که بسته به نوع نقص این چهار سطح را تشکیل می دهیم. این دسته بندی به این دلیل است که مقادیر آموزنده تر و دقیق تر برای آزمون جعبه سیاه طبقه بندی شود.
مرحله آخر نیز مجددا از روش های مرحله دوم و سوم به منظورکالیبره کردن و بررسی مجدد استفاده می کند.

حال پس از اتمام مراحل بالا و طبقه‌بندی معایب نرم‌افزار، می‌توان الگوهای آن دست از نقص‌هایی که مکررا رخ می‌دهند را استخراج نمود. استخراج دستی الگوهای نقص مکرر با وجود نقص ها با تعداد بالا کار بسیار دشواری می باشد. بنابراین ایجاد الگویی برای استخراج الگوهای مکرر نقص از مخازن نقص نقش مهمی در تشخیص و تحلیل نرمافزار دارد. در استخراج الگوهای مکرر، مجموع نقص های مکرر و استخراج وابستگی بین نقص ها دو موضوع بسیار مهم می باشد. استخراج الگوهای نقص بر اساس روش طبقه بندیODC-BD می باشد. در مجموع ۳۰۰ نوع نقص برای تست جعبه سیاه وجود دارد[۸]. به طورکلی یک نقص جعبه سیاه به یک تابع یا چندین تابع برنامه وابسته است، استخراج قوانین وابستگی برای حذف ارتباط بین نقص های یک تابع خاص مورد استفاده می باشد. یک قانون وابستگی بصورت A => B می باشد. به طورکلی استخراج قوانین وابستگی با استفاده از الگوریتم Aprior(R. Agrawal and R. Srikant,1994)و FP-growth(J. Han, J. Pei, and Y. Yin) می باشد.

اساس الگوریتم Apriori و FP-growth یک K-itemset است که همه آیتم های این مجموع مکررا رخ داده است. این به این معنی است که الگوریتم ابتدا برای بار اول مخزن نقص ها را اسکن کرده و یک itemset ایجاد می کند و سپس مجدد مخزن نقص را اسکن می کند و 1-itemset ایجاد می شود و این روند تکراری را k مرحله ادامه می دهیم تا هیچ نقص مکرری یافت نشود سپس ۱۰ نقص بالای itemset را ذخیره می کنیم این ۱۰ نقص بالای لیست بدان معناست که این نقص ها به احتمال زیاد رخ می دهند، با استفاده از این روش و اسکن مکرر مخزن نقص ها و استخراج ده نقص پرتکرار و شناسایی آنها می توان با سرعت بیشتری موارد آزمون را طراحی و تست را انجام داد همچنین شناسایی قوانین وابستگی به تست کنندگان کمک می کند که ارتباط بین نقص ها را شناسایی و حذف کند(N. Li, Z. Li, and L. Zhang,2010).

نتیجه گیری

تست نرم‌افزار اساسا به منظور شناسایی خطاها و اشکالات نرم‌افزار، و از بین بردن تمامی آن‌ها مورد استفاده قرار می‌گیرد. تست جعبه سیاه بدون توجه به ساختار داخلی برنامه فقط به بررسی صحت خروجی برنامه تمرکز دارد، که با استفاده از ایجاد موارد آزمون برای انجام تست و همچنین طبقه بندی و دسته بندی براساس نوع نقص با استفاده از روش طبقه بندی ODC-BD، تمرکز و دقت برروی برخی نقص های پرتکرار بیشتر و در نهایت خطاهای کشف شده برنامه کمتر می شود.
هدف از این مقاله ایجاد دید کلی و روشنی از مبحث تست نرم‌افزار، روش های ایجاد موارد آزمون، آزمون تست جعبه سیاه و شناسایی نقص های آن در اختیار خوانندگان قرار دهیم. امید است بتواند راهنمای مناسبی را در این زمینه فراهم آورد.

مراجع

]1[K. K. Mohan, A.K.Verma, and A. Srividya, "Software Reliability Estimation Through Black Box and White Box Testing at Prototype Level," *IEEE,* p. 6, .2010

]2[K. Kaur and S. Sharma, "A Survey on Software Testing," *International Journal of Emerging Trends & Technology in Computer Science,* p. 4, .2015

]3[M. E. Khan and F. Khan, "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques," *International Journal of Advanced Computer Science and Applications,,* p. 4, .2012

]4[T. Hussain and D. S. Singh, "A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing," *IJAPRR International Peer Reviewed Refereed,* p. 8, .2015

]5[L. Williams, "Testing Overview and Black-Box Testing Techniques ",p. 26, .2006

]6[M. A. Khan and M. Sadiq, "Analysis of Black Box Software Testing Techniques: A Case Study," *IEEE,* p. 5, .2011

]7[N. Li, Z. Li, and a. X. Sun, "Classification of Software Defect Detected by Black-box Testing: An Empirical Study," *IEEE* ,p. 7, .2010

]8[N. Li, Z. Li, and L. Zhang, "Mining Frequent Patterns from Software Defect Repositories for Black-box Testing," *IEEE,* p. 4, .2010

]9[R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," presented at the Conference Santiago, Chile, .1994

]10[J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *Natural Sciences and Engineering Research Council of Canada*.