УДК 621.396.26

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РАДІОЕЛЕКТРОННИХ ЗАСОБІВ

Мірських Г.О., к.т.н., доцент; Реутська Ю.Ю., асистент
Національний технічний університет України "Київський політехнічний інститут", М. Київ, Україна

Тестування лише демонструє наявність помилок у програмних продуктах, але не показує відсутність помилок, тобто не є і не може бути пов'язане з мірою якості цих програмних продуктів

Невідомий автор

Сучасний радіоелектронний засіб (РЕЗ) – складна система, робота якої відбувається згідно визначених алгоритмів. Реалізація цих алгоритмів здійснюється відповідним програмним забезпеченням – програмним продуктом (ПП), що на сьогодні розглядається як невід'ємна складова РЕЗ. Надійність та якість ПП в значній мірі визначають споживчі характеристики РЕЗ в цілому, а отже підлягають вимірюванню, контролю, прогнозуванню та забезпеченню на протязі усього життєвого циклу РЕЗ. В той же час ПП, як об'єкт дослідження, суттєво відрізняється від інших складових електронного апарату, а отже відрізняється й сенс категорій, яким визначається споживча якість такого об'єкту.

Особливості визначення категорій якості та надійності щодо програмного забезпечення РЕЗ

У звичному для спеціалістів з радіоелектроніки сенсі категорія якості пов'язується з відсутністю (точніше з не виявленням) у об'єкті дефектів під час проведення відповідних контрольних заходів, а категорія надійності – зі спроможністю проявляти об'єктом відповідні споживчі властивості протягом визначеного періоду, тобто знову ж таки з відсутністю проявів дефектів протягом визначеного періоду. Характерним при цьому є те, що виправлення дефекту, що проявляється на виході виробничого циклу та/або під час експлуатації, не змінює споживчі властивості об'єкту (випадки щодо дороблення або модернізації об'єкту за результатами аналізу проявлення дефекту не розглядаються). Специфікою розглянутих категорій якості та надійності РЕЗ є наявність відповідної міри, а отже і відповідних методів та засобів для їх визначення. Слід відзначити, що дефекти РЕЗ, які проявляються на стадіях їх виробництва та експлуатації, зазвичай, пов'язані з відповідними дефектами матеріалів, комплектуючих, з порушеннями тех-

нологічного процесу тощо і не пов'язані (або переважно не пов'язані) з помилками, допущеному під час проектування.

Складність, багатофункціональність, залежність результатів роботи ПП від наявних даних на вході тощо, не дозволяє ввести міру для категорій "якість" та "надійність", а отже і не дозволяє визначити ці поняття в формальнологічному сенсі, нормувати та вимірювати їх. Зазвичай, вважають[1-5], що в ПП є дефекти, якщо вона не виконує те, що доречно від неї очікувати споживачеві. Крім того, дефекти, що проявляються на стадій експлуатації ПП, є *помилками*, припущеними на стадії проектування (розроблення), а отже ліквідація дефекту у ПП є, за своєю суттю, виправленням раніше припущеної помилки. Тобто виправлення ПП за результатами виявлення помилки призводить до появлення нового ПП, з новими властивостями. При цьому слід враховувати, що помилки, припущені під час розроблення ПП, можуть за визначених умов нівелювати одна одну, а отже виправлення однієї з таких помилок може призвести до погіршення роботи ПП в цілому.

Характерною ілюстрацією відмінності понять надійності апаратної та програмної складових РЕЗ можуть бути відомі залежності інтенсивності проявлення дефектів під час експлуатації (див. рис. 1).

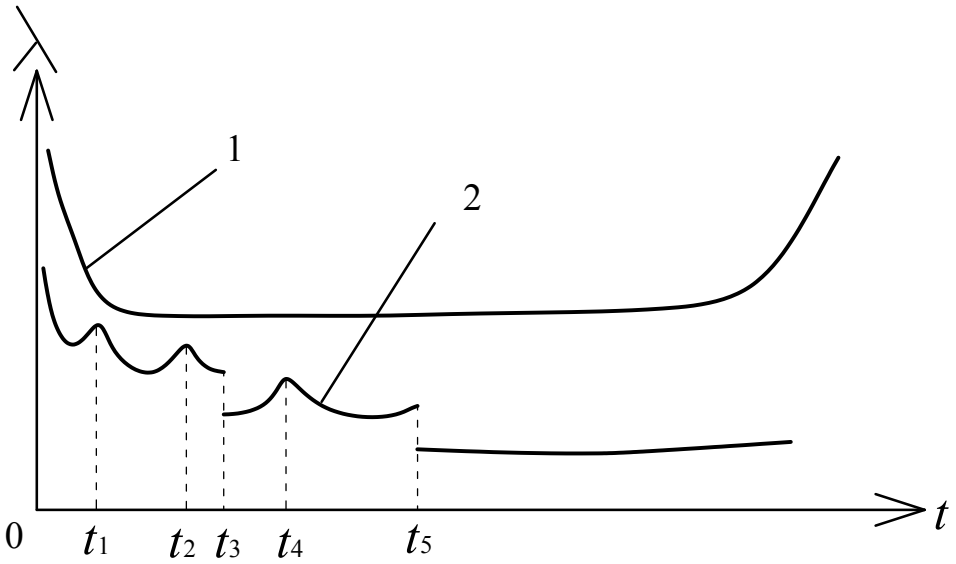

Рис. 1.

Кривою 1 (рис. 1) наведена відома залежність інтенсивності відмов апаратної складової РЕЗ (за умови можливості ремонту) від часу експлуатації. Характерним для цієї залежності є відрізок часу стабільної експлуатації, коли кількість відмов незначна, а їх появлення в цілому прогнозується з визначеною ймовірністю.

Кривою 2 (рис. 1) показана залежність від часу експлуатації ПП (програмної складової РЕЗ). При цьому за віссю ординат наведена деяка абст-

рактна величина, що характеризує правильність роботи ПП та пов'язана з інтенсивністю проявлення помилок. Видно, що протягом часу експлуатації вказана залежність має "сплески" та "перепади" що відповідають моментам виявленню у ПП помилок. При цьому виправлення помилок у моменти часу t_1, t_2, t_4 відповідає ситуації, коли усунення цих помилок з визначеною часткою умовності можна вважати модернізацією ПП. "Перепади" у моменти часу t_3, t_5 пов'язані з помилками, що легко ліквідуються. З часом таких "сплесків" та "перепадів" стає менше і, коли кількість помилок, що залишились у ПП стає вельми малою (а це може статися і наприкінці експлуатаційного періоду РЕЗ) наведена крива стабілізується.

Час проявлення помилок у ПП, звичайно, величина випадкова, пов'язана з умовами експлуатації ПП (які, звичайно корінним чином відрізняються від умов експлуатації РЕЗ), а саме з вихідною інформацією (характером, порядком та швидкістю її надання тощо), послідовністю виконання тих чи інших фрагментів програми, кваліфікацією та/або психологічним станом осіб, що причетні до експлуатації РЕЗ, тощо. Як видно з характеру наведеної кривої, що виправлення чергової помилки не дає впевненості в підвищенні якості ПП.

Наведені особливості категорій, які характеризують якість ПП, показують, що забезпечення його безпомилкової роботи під час експлуатації можна досягти виключно відповідною організацією процесу проектування, використанням таких технологій, котрі б забезпечували виявлення максимальної кількості допущених розробниками ПП помилок ще на стадії проектування.

Одним з найефективніших засобів, здатних виявити дефекти ПП, на сьогодні вважається *тестування*. За визначенням тестування – це процес виконання програми (або її частини) з наміром знайти помилки. Характерна особливість цього процесу – заздалегідь відомий результат, що має бути отриманий за умови правильного функціонування ПП. Тестування не є контролем або випробуванням ПП в повному сенсі цих категорій. По відношенню до ПП контроль слід розуміти як спробу знайти помилки, використовуючи тестування ПП в визначених штучних (створених спеціально для проведення тестування) умовах, а випробування ПП можна визначити як процес пошуку помилок за умови виконання програми в заданому реальному середовищі, при цьому можуть виконуватися як тестові, так і реальні задачі. Відмітимо, що в обох наведених визначеннях мова йде саме про пошук помилок, зважаючи на те, що на сьогодні не існує способу доведення відсутності помилок у ПП.

Слід зауважити, що процес тестування, за визначенням не співпадає з *ладнанням* ПП, яке спрямоване на встановлення точної природи помилки (виявленої за результатами тестування) та її виправлення. Тобто результати тестування ПП є вихідними даними для здійснення його ладнання.

Тестування, як основний засіб забезпечення якості ПП

Наведене показує, що всі категорії, за допомогою яких можна в тій чи іншій мірі характеризувати якість ПП, пов'язані з тестуванням – з розробленням відповідних тестових задач (тестів) та їх розв'язанням за допомогою даного ПП на різних стадіях його життєвого циклу.

З метою мінімізації кількості помилок, що "закладаються" у ПП під час його проектування, комплекс пов'язаних з розробленням та реалізацією тестів заходів охоплює всі етапи цієї стадії життєвого циклу ПП та включає:

- постановку задачі для розроблення тестових завдань;
- проектування (написання) тестових завдань;
- тестування тестів на предмет їх відповідності поставленим задачам;
- реалізацію процесу тестування (виконання тестів);
- вивчення результатів тестування.

Звичайно, чи не найважливішою складовою наведеного переліку є задача проектування самих тестів. З методологічної точки зору можна виділити два напрямки розв'язання цієї задачі [4-6], які на практиці, зазвичай, використовуються паралельно, в залежності від цілей тестування та об'єму програми або її частини, для яких розробляється відповідний тест..

Перший методологічний напрямок стосується безпосередньо тексту програми, для якої саме й проектується тест. Проектування тесту спрямоване на виявлення проблем у внутрішній структурі програми. відбувається Ця методологія отримала назву "*білого ящика*". Загальна задача тестування при цьому – забезпечити перевірку кожного кроку алгоритму програми. Головна перевага всіх типів стратегій тестування, заснованих на методології "білого ящика" – врахування під час тестування структури (логіки) всієї програми, що, зазвичай, полегшує виявлення помилок навіть у випадку, коли специфікації ПП недостатньо визначені або неповні. Повним тестуванням програми за такою методологією слід вважати перебирання всіх можливих шляхів, за якими у ПП буде отримана відповідна інформація на виході при заданих вхідних даних, що є задачею, практична реалізація якої неможлива, адже навіть для середніх по складності програм кількість таких шляхів сягатиме десятків тисяч. Як правило, розроблення тестів та сам процес тестування за методом "білого ящика" здійснюється розробниками ПП на стадії його ладнання.

Другий - не передбачає знання тексту програми, але потребує чіткого визначення специфікацій (структур даних на вході та виході) та інформації щодо загальної схеми роботи ПП. Така методологія, що базується на зовнішніх функціях ПП, отримала назву "чорного ящика" (функціональне тестування). Тестування "чорного ящика" полягає в пошуку функцій ПП, що відсутні або неправильно виконуються, з метою визначення наскільки споживчі властивості ПП відповідають заданим вимогам. Функціональне тестування, зазвичай підтверджує правильність даних на вході та виході

ПП. Таке тестування часто називають тестуванням користувача. Методологія "чорного ящика" на сьогодні є найпоширенішою при розробленні тестів для фінішного тестування ПП, але за такий спосіб неможливо виявити помилки, що взаємно нівелюються, та помилки що зустрічаються рідко, адже, як правило, неможливо на стадії тестування ПП передбачити всі можливі комбінації вхідних та вихідних даних, з якими дана програма працюватиме під час експлуатації.

Інтеграційне тестування, як засіб забезпечення якості складних ПП

На етапі збирання до єдиного програмного комплексу програмних модулів, на сьогодні використовують дві стратегії [8-11]:

- монолітності, згідно якої всі програмні модулі об'єднуються одночасно;
- інкрементальності, яка характеризується покроковим (помодульним) нарощуванням програмного комплексу.

Цим стратегіям відповідають аналогічні за назвами стратегії тестування програмних комплексів в процесі їх створення.

Особливості стратегії монолітності полягають в необхідності розроблення спеціальних драйверів та/або заглушок, з метою заміни модулів, що не розроблені до моменту початку процесу їх інтеграції та тестування. Це потребує додаткових (часто значних) витрат. Крім того така стратегія пов'язана зі складністю ідентифікації помилок, що проявляються в просторі зібраної частини програмного комплексу.

Інкрементальна стратегія збирання та тестування програмних комплексів, пов'язана з суттєво меншими трудовитратами щодо ідентифікації помилок, за рахунок поступового нарощування об'єму програми, яка тестується.

На сьогодні існує великий вибір можливих підходів щодо реалізації методології тестування, з метою забезпечення якості складних програмних комплексів на стадії їх проектування. В узагальненому вигляді всі ці підходи можна розглядати як варіанти шести базових методів, що складають основу методології *інтеграційного тестування* ПП - тестування, з метою виявлення помилок в процесі інтеграції програмних модулів в складний програмний комплекс, тобто з метою виявлення дефектів, пов'язаних з помилками в реалізації та інтерпретації інтерфейсного зв'язку між модулями програми.

Метод "великого стрибка"

Метод "великого стрибка" на сьогодні є чи не найпоширенішим методом інтеграції програмних модулів. Відповідно до цього методу перед інтеграцією кожен модуль тестується автономно. Після тестування всі модулі інтегруються в систему одночасно.

Метод "великого стрибка" у порівнянні з іншими методами має чимало недоліків і мало переваг:

- заглушки та драйвери необхідні для кожного модулю;
- модулі не інтегруються до останнього моменту, а це означає, що протягом тривалого часу суттєві недоліки та помилки у їх спряженні залишаються невиявленими;
- одночасна інтеграція всіх модулів суттєво ускладнює ладнання програмного комплексу.

Метод "великого стрибка" рекомендується до використання для невеликих за об'ємом програм, за умови їх проектування досвідченими фахівцями. Однак для значних за об'ємом програм цей метод зазвичай є пагубним.

Тестування за методом "знизу вверх"

Тестування за методом "знизу вверх" передбачає збирання та тестування програми "знизу вверх". Лише модулі найнижчого рівня ("термінальні" модулі, що не передають інформацію до інших модулів, а працюють виключно в режимі приймання інформації) тестуються автономно. Після того як тестування цих модулів завершене, виклик їх має бути гарантованим. Після цього тестуванню піддаються модулі більш високого рівня, що безпосередньо викликають вказані вже перевірені модулі. Ці модулі тестуються не автономно, а разом із вже перевіреними модулями більш низького рівня. За такий спосіб процес повторюється до тих пір, поки не буде зібраним весь програмний комплекс. Звичайно, використання цього методу тестування потребує розроблення спеціального драйвера для кожного програмного модулю, адже необхідно подавати тести відповідно зі спряженням модуля, що тестується.

Тестування за методом "зверху вниз"

Тестування за методом "зверху-вниз" передбачає збирання та тестування програми "зверху вниз". За таких умов ізольовано тестується лише головний модуль. Надалі один за другим підключають модулі, що безпосередньо викликаються головним модулем, і тестується отримана комбінація. Цей процес повторюється до тих пір, поки не будуть зібрані та перевірені всі модулі. При цьому для імітації функцій модулів, що ще не підключені до комплексу, але мають викликатися модулями, що підключаються та тестуються на даному етапів, розробляються спеціальні *модулі-заглушки,* котрі моделюють функції цих відсутніх модулів. Задача розроблення модулів-заглушок, зазвичай виявляється вельми складною, що є одним із суттєвих недоліком методу тестування "зверху вниз. Поруч з цим слід відмітити й суттєві переваги цього методу, який забезпечує суміщення тестування самого модулю, елементів його з'єднання (спряження) з іншими модулями програмного комплексу та часткове тестування зовнішніх функцій. Цей підхід є найпривабливіший при розробленні складного програмного комплексу, особливо за умови наявності сумнівів щодо можливості його реалізації в цілому (зокрема за ймовірної наявності суттєвих дефектів у

формулюванні вихідних даних до проекту).

Комбіновані методи тестування

Ефективність використання кожного з наведених методів тестування ПП пов'язана з його структурою, організаційними та фінансовими умовами реалізації проекту. Істотно постає питання щодо можливості комбінації окремих методів, з метою розширення області їх ефективного використання та нівелювання притаманних їм недоліків. Така комбінація окремих методів тестування призводить до їх відповідної модифікації, що нерідко стає у нагоді.

Модифікація методу "зверху вниз"

Суттєвим недоліком збирання та тестування програми за методом "зверху вниз" в наданій вище інтерпретації є неможливість тестувати деякі логічні умови, наприклад помилкові ситуації або захищені перевірки. Цей метод робить надзвичайно складною, або взагалі неможливою перевірку специфічних ситуацій у деякому модулі, якщо програма працює з цим модулем лише в обмеженому контексті (це означає, що цей модуль ніколи не отримає достатньо повний набір вхідних значень).

З метою запобігання невизначеності в подібних ситуаціях необхідно кожен подібний модуль перед підключенням до вже зібраної програми піддавати автономному тестуванню, що і визначає сутність модифікації методу "зверху вниз". При цьому, однак, постають додаткові труднощі з розробленням відповідних драйверів та заглушок.

Метод сандвіча – комбінація методів "зверху вниз" та "знизу вверх"

Тестування за методом сандвіча є компромісом спрямованим на використання переваг методів тестування "знизу вверх" та "зверху вниз". Під час використання цього методу одночасно починають тестування (а відповідно й збирання) програмного комплексу й "знизу вверх" й "зверху вниз". За такий спосіб інтеграції модулів "точка зустрічі" знаходиться у середині програмного комплексу і має бути визначена заздалегідь, на підставі прискіпливого аналізу структури всієї програми. Метод сандвіча, зазвичай, використовується під час проектування значних за об'ємом програм.

Метод сандвіча зберігає такі переваги методів "знизу вверх" та "зверху вниз", як початок інтеграції системи на самому ранньому етапі проектування. Тобто вже на ранніх стадіях збирання програмного комплексу проектувальники отримують і працюючий каркас програми і можливості тестувати модулі низького рівня. Однак під час використання методу сандвіча виникає та сама проблема (неможливість тестування окремих модулів), що й при використанні методу "зверху вниз", хоча й не в такій жорсткій формі. Використання для частини програми, що тестується за методом сандвіча, методу "знизу вверх" розв'язує цю проблему для модулів нижніх рівнів, але вона залишається для нижньої половини верхньої частини програми.

Ці недоліки нівелюються відповідною модифікацією методу сандвіча.

Модифікація методу сандвіча

В модифікованому методі сандвіча модулі нижніх рівнів також тестуються строго знизу вверх. А ось модулі верхніх рівнів спочатку тестуються ізольовано й лише після цього збираються за методом "зверху вниз". Таким чином модифікований метод сандвіча є додатковим компромісом між методами тестування "знизу вверх" та "зверху вниз".

Порівняльна характеристика методів тестування ПП

Комплексну оцінку прийнятого методу тестування ПП з урахуванням надійності, часу та витрат, яких потребує його розроблення, можна оцінити за відповідними критеріями, як показано у таблиці.

Таблиця

Критерії порівняння	Метод тестування програмного продукту					
	"Знизу вверх"	"Зверху вниз"	"Зверху вниз" (модиф.)	Метод "великого стрибка"	Метод сандвіча	Метод сандвіча (модиф.)
Отримання зібраної програми	Рано	Рано	Рано	Пізно	Рано	Рано
Час до появи працюючого варіанту програми	Пізно	Рано	Рано	Пізно	Рано	Рано
Потреба драйверів	Так	Ні	Так	Так	Частково	Так
Потреба заглушок	Ні	Так	Так	Так	Частково	Частково
Паралелізм в роботі над програмою	Середній	Низький	Середній	Високий	Середній	Високий
Можливості щодо тестування окремих шляхів	Легко	Важко	Легко	Важко	Середнє	Легко
Можливості щодо планування та контролю послідовності розроблення програми	Легко	Важко	Важко	Легко	Важко	Важко

Перший критерій визначає відрізок часу від початку тестування до моменту інтеграції модулів, оскільки це суттєво впливає на виявлення помилок, пов'язаних з обміном інформацією між модулями.

Другий — визначає відрізок часу від початку тестування до моменту створення перших працюючих "скелетних" версій програми, адже саме тут можуть проявитися головні дефекти проектування. Вочевидь, що чим раніше з'явиться така версія, тим менше часу й коштів доведеться витратити на здобуття інформації щодо суттєвих недоліків проекту.

Третій та четвертий - визначають міру "платні", за використання відповідних методів тестування, що визнані менеджерами проекту найбільш придатними під час розроблення даного програмного комплексу.

П'ятий — визначає можливості щодо паралельного розроблення окремих складових програми. Звичайно, паралельна робота над окремими складовими програми зменшує час проектування, але потребує більшої кількості учасників проекту. Пріоритетними вважаються методи, що дозволяють отримати можливості паралельної роботи над програмою на початкових етапах тестування.

Шостий критерій пов'язаний з можливостями вибраного методу тестування щодо перевірки будь-яких шляхів або умов, якими наповнена конкретна програма

Сьомий - характеризує складність планування, реалізації заходів з контролю над ходом робіт та управління цими роботами протягом всього часу тестування. Адже будь-який процес, який важко піддається контролю, яким важко управляти, часто веде до упущень, помилкам, до не раціонального витрату часу та коштів тощо.

Висновки

Специфіка програмного продукту, як складової РЕЗ (апаратно-програмного комплексу), не дозволяє ввести для нього одиниці виміру таких показників як рівень якості та надійність, а отже унеможливлює оцінювання та нормування цих показників. Для забезпечення якості складного ПП необхідно використовувати відповідні методи його проектування, що тісно пов'язані з поточним тестуванням під час виконання проектних робіт. Вибір методу тестування визначається сукупністю факторів, серед яких виділяються структура та об'єм програми, людські та фінансові ресурси. Комбінація різних за методологією методів надає чимало переваг у реалізації вільних від помилок програмних комплексів.

Література

1. Майерс Г. Надежность программного обеспечения. - М.: Мир, 1980.
2. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. - М.: Мир, 1985.
3. Фокс Дж. Программное обеспечение и его разработка. - М.: Мир, 1985.
4. Жоголев Е.А. Введение в технологию программирования. - М.: "ДИАЛОГ-МГУ", 1994.
5. Майерс Г. Искусство тестирования программ. - М.: Финансы и статистика, 1982.
6. Котляров В.П. Основы тестирования программного обеспечения. Интернет-университет информационных технологий – INTUIT.ru, 2006 г.
7. Смагин В.А., Солдатенко В.С., Кузнецов В.В. Моделирование и обеспечение надёжности программных средств – СПб.: ВИКУ им. А.Ф. Можайского, 1999.
8. Канер Ф.Н. Тестирование программного обеспечения. М.:, Изд. "ДиаСофт", 2000, 378 с.
9. Бахтизин В. В., Глухова Л. А. Стандартизация и сертификация программного обеспечения — Минск: БГУИР, 2006. — 200 с.
10. Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование: Изд. дом "Вильямс" / Сер. ин-та качества программного обеспечения — 374с.
11.Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. М.: БИНОМ, 2008, 368 с.

Мірських Г.О., Реутська Ю.Ю. ***Порівняльний аналіз методів тестування програмного забезпечення радіоелектронних засобів.*** *Проведено аналіз понять якості та надійності програмних продуктів, що є складовою частиною радіоелектронних засобів. Наведені основні методи тестування програмних продуктів, що використовуються під час проектування апаратно-програмних комплексів з метою забезпечення їх якості та надійності. Розглянуто тестування за методологією "чорного ящика" та "білого ящика", методи інтеграційного тестування "знизу-вверх" та "зверху вниз", а також різні модифікації цих методів. Наведені ефективні критерії, що дозволяють здійснювати вибір методів тестування програм з урахуванням як їх структури, так і відповідних організаційних та фінансових чинників, що впливають на якість реалізації процесу проектування.*

Ключові слова: *програмний продукт, тестування програм, якість програм, надійність радіоелектронних засобів*

Мирских Г.А., Реутская Ю.Ю. ***Сравнительный анализ методов тестирования программного обеспечения радиоэлектронных средств****. Проведен анализ понятий качество и надежность программных продуктов, которые являются составными частыми радиоэлектронных средств. Приведены основные методы тестирования программных продуктов, которые используются в процессе проектирования аппаратно-программных комплексов, с целью обеспечения качества и надежности. Рассмотрено тестирование в соответствии с методологией "черного ящика" и "белого ящика", методы интеграционного тестирования "снизу вверх" и "сверху вниз", а также различные модификации этих методов. Приведены эффективные критерии, которые позволяют выбрать метод тестирования программ с учетом их структуры и организационно-финансовых факторов, которые влияют на качество реализации процесса проектирования.*

Ключевые слова*: программный продукт, тестирование программ, качество программ, надежность радиоэлектронных средств.*

Mirskikh G.A., Reutskaya J.J. ***Comparative analysis of methods for testing software of radio-electronic equipment.*** *The analysis of the concepts of quality and reliability of software products that are part of the radio-electronic equipment is making. Basis testing methods of software products that are used in the design of hardware and software systems, to ensure quality and reliability are given. We consider testing in accordance with the methodology of the "black box" and "white box" methods of integration testing from the bottom up and top down, as well as various modifications of these methods. Efficient criteria that allow you to select the method of testing programs based on their structure and organizational and financial factors that affect the quality of the implementation of the design process are leaded.*

Keywords: *software, software testing, software quality, reliability of radio electronic equipment.*