УДК 355.233.1.005

І.О. Романенко[1], К.С. Смеляков[2], І.В. Рубан[2], С.В. Алексєєв[2], В.В. Калачова[2], О.П. Бабенко[3]

[1] *Національний університет оборони України, Київ*
[2] *Харківський університет Повітряних Сил ім. І. Кожедуба, Харків*
[3] *Міністерство оборони України, Київ*

МЕТОДИКА ІНТЕГРАЛЬНОЇ ОЦІНКИ СТУПЕНЯ ТЕСТОВАНОСТІ ТА СТРУКТУРНИЙ АЛГОРИТМ ТЕСТУВАННЯ СПЕЦІАЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Оцінювання рівня підготовленості органів військового управління, частин і підрозділів, особового складу Збройних Сил України в більшості здійснюється неавтоматизованим способом, тому актуальною задачею сьогодення є удосконалення процедури оцінювання, забезпечення своєчасності контрольних заходів, досягнення максимальної об'єктивності, підвищення економічної ефективності за рахунок автоматизації процесу підготовки. При цьому одним із ключових питань постає тестування спеціального програмного забезпечення, що розробляється. Запропоновані методика інтегральної оцінки ступеня тестованості спеціального програмного забезпечення та структурний алгоритм його тестування гарантують досягнення високого рівня якості програмного продукту.

Ключові слова: *спеціальне програмне забезпечення, управляючий граф програми, ступень тестованості, набір трас, критерій шляхів.*

Вступ

Постановка проблеми. Сучасний розвиток військової науки та використання новітніх електронних і комп'ютерних технологій в озброєнні та військовій техніці вимагає високого професіоналізму від військовослужбовців, а перехід Збройних Сил України до професійної армії на контрактній основі – якісної та

гнучкої системи підготовки висококваліфікованих кадрів, складовою якої є підсистема оцінювання рівня підготовленості органів військового управління, частин і підрозділів, особового складу до виконання завдань за призначенням [1].

Впровадження новітніх інформаційних технологій у військову практику оцінювання має на меті: удосконалення процедури оцінювання, забезпечення своєчасності контрольних заходів, досягнення максимальної об'єктивності, підвищення економічної ефективності за рахунок автоматизації процесу підготовки взагалі. При цьому одним із ключових питань постає тестування розробленого спеціального програмного забезпечення з метою визначення його відповідності задачам, що ним розв'язуються, а також оцінки достовірності отриманих результатів.

Тестування є одним із найбільш сталих способів забезпечення якості розробки програмного забезпечення і входить до набору ефективних засобів сучасної системи контролю якості програмних продуктів [2, 3]. Воно дозволяє виявити й усунути помилки у програмному продукті. Якість програмного продукту характеризується набором властивостей, що визначають, наскільки продукт задовольняє вимогам зацікавлених сторін, таких як замовник, користувач, розробники і тестувальники продукту. Кожен із учасників може мати різне уявлення про продукт і про те, наскільки він хороший чи поганий, тобто про те, наскільки висока якість продукту. Таким чином, постановка задачі забезпечення якості продукту виливається в завдання визначення множини критеріїв якості з наступним знаходженням оптимального рішення, що задовольняє цим критеріям [4, 5].

Застосування математичного апарату системного аналізу та моделювання при виконанні тестування створеного програмного продукту відкриває один з можливих шляхів досягнення високого ступеня тестованості програмного забезпечення.

Аналіз останніх досліджень і публікацій. Відомості щодо питання впровадження новітніх інформаційних технологій у військову практику оцінювання докладно надано в [1]. Дослідження процедури тестування та визначення якості ПЗ наведено в [3 – 5].

Принципи та особливості відомих методик оцінки ступеня тестованості програмного забезпечення наведено в [6 – 8].

Метою статті є представлення аналізу основних критеріїв та методів тестування програмних продуктів та висвітлення розроблених на їх основі методики інтегральної оцінки ступеня тестованості та структурного алгоритму тестування програмного забезпечення.

Виклад основного матеріалу досліджень

Сутність методики інтегральної оцінки ступеня тестованості програмного забезпечення полягає в наступному:

Крок 1. Вибір критерію C і приймальної оцінки тестування програмного проекту –L.

Основними критеріями, що застосовуються при тестуванні програмних модулів є *структурні*, *функціональні та стохастичні*.

Ідеальний критерій тестування повинен бути:

– достатнім, тобто показувати, коли деяка кінцева множина тестів є достатньою для тестування даної програми;

– повним, тобто в разі помилки повинен існувати тест з множини тестів, що задовольняють критерію, який розкриває помилку;

– надійним, тобто будь-які дві множини тестів, що задовольняють йому, одночасно повинні розкривати або не розкривати помилки програми.

– повинен легко перевірятися у обчислювальному сенсі на тестах.

Для нетривіальних класів програм в загальному випадку не існує повного і надійного критерію, що залежить від програм або специфікацій. Тому при розробці спеціального програмного забезпечення для цілей тестування доцільно використовувати сукупність окремих критеріїв, основними з яких є: структурні, функціональні, стохастичні.

Структурні критерії використовують модель програми у вигляді «білого ящика», що передбачає знання тексту програми або специфікації програми у вигляді потокового графа управління (УГП). Структурна інформація зрозуміла і доступна розробникам підсистем і модулів програми, тому даний клас критеріїв часто використовується на етапах модульного й інтеграційного тестування (Unit testing, Integration testing). Структурні критерії базуються на основних елементах УГП, операторах, гілках і шляхах.

Умова критерію тестування команд (критерій C0) – набір тестів у сукупності має забезпечити проходження кожної команди не менше одного разу. Це слабкий критерій; він, як правило, використовується у великих програмних системах, де інші критерії застосувати неможливо.

Умова критерію тестування гілок (критерій C1) – набір тестів у сукупності має забезпечити проходження кожної гілки не менше одного разу. Це досить сильний і при цьому економічний критерій, оскільки множина гілок у програмі, що тестується, звичайно не така велика. Цей критерій часто використовується в системах автоматизації тестування.

Умова критерію тестування шляхів (критерій C2) – набір тестів у сукупності має забезпечити проходження кожного шляху не менше 1 разу. Якщо програма містить цикл (особливо з неявно заданими числом ітерацій), то число ітерацій обмежується константою (часто – 2, або числом класів вихідних шляхів).

Функціональний критерій – це найважливіший для програмної індустрії критерій тестування. Він забезпечує, перш за все, контроль ступеня виконан-

ня вимог замовника в програмному продукті. При функціональному тестуванні переважно використовується модель «чорного ящика». До видів функціональних критеріїв відносять: тестування пунктів специфікації (перевірку кожного пункту специфікації не менше одного разу); тестування класів вхідних даних (перевірка представника кожного класу вхідних даних не менше одного разу); тестування правил (перевірка кожного правила, якщо вхідні і вихідні значення описуються набором правил деякої граматики); тестування класів вихідних даних (перевірка представника кожного вихідного класу, за умови, що вихідні результати заздалегідь класифіковані, причому окремі класи результатів враховують, у тому числі, обмеження на ресурси або на час (time out)); тестування функцій (перевірка кожної дії, що реалізує модуль, який тестується, не менше одного разу).

Стохастичне тестування застосовується при тестуванні складних програмних комплексів, коли набір детермінованих тестів (X, Y) має величезну потужність. До *критеріїв стохастичного тестування* відносять статистичні критерії закінчення тестування та критерії оцінки швидкості виявлення помилок [6, 7].

Вибір приймальної оцінки тестування програмного проекту L (рівень відтестованості), як правило, задається у вимогах до програмного продукту та знаходиться в межах ($0 \le L \le 1$).

Крок 2. ***Побудова управляючого графу програми.***

Для оцінки ступеня тестування часто використовується управляючий граф програми – УГП. Наприклад, УГП багатокомпонентного об'єкта G містить в собі два компоненти $G1$ і $G2$, УГП яких розкриті (рис. 1).

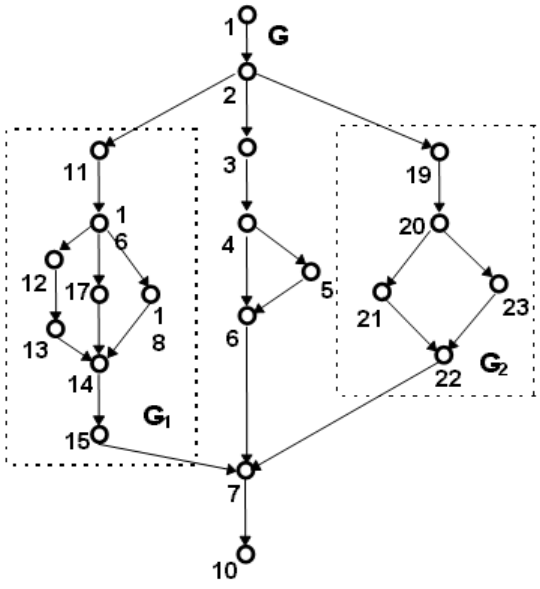

Рис. 1. Плоска модель УГП компонента G

У результаті УГП компонента G має такий вигляд, як якщо б компоненти $G1$ і $G2$ в його структурі спеціально не виділялися, а УГП компонентів $G1$ і $G2$ були вставлені в УГП G. Для тестування компонента G відповідно до критерію шляхів потрібно прогнати тестовий набір, що покриває відповідний набір трас графа G.

Крок 3. ***Модульне тестування й оцінка степеня тестованості на модульному рівні.***

Модульне тестування є тестуванням програми на рівні окремих модулів, функцій або класів. Мета цього тестування полягає у виявленні локалізованих у модулі помилок в реалізації алгоритмів, а також у визначенні ступеня готовності системи до переходу на наступний рівень розробки. Модульне тестування проводиться за принципом «білого ящика», тобто ґрунтується на знанні внутрішньої структури програми, і часто включає ті чи інші методи аналізу покриття коду [7, 8].

Модульне тестування зазвичай передбачає створення навколо кожного модуля певного середовища, що включає заглушки для всіх інтерфейсів модуля, що тестується. Деякі з них можуть використовуватися для подачі вхідних значень, інші для аналізу результатів, присутність третіх може бути продиктована вимогами, що накладаються компілятором і збирачем. На рівні модульного тестування найпростіше виявити дефекти, пов'язані з алгоритмічними помилками і помилками кодування алгоритмів, типу роботи з умовами та лічильниками циклів, а також з використанням локальних змінних і ресурсів. Помилки, пов'язані з невірним трактуванням даних, некоректною реалізацією інтерфейсів, сумісністю, продуктивністю тощо зазвичай пропускаються на рівні модульного тестування і виявляються на більш пізніх стадіях тестування.

Саме ефективність виявлення тих чи інших типів дефектів повинна визначати стратегію модульного тестування, тобто розстановку акцентів при визначенні набору вхідних значень.

Будучи за способом виконання структурним тестуванням або тестуванням «білого ящика», модульне тестування характеризується ступенем, в якій тести виконують або покривають логіку програми (вихідний текст). Тести, пов'язані зі структурним тестуванням, будуються за такими двома принципами.

Принцип 1. На основі аналізу потоку управління. В цьому випадку елементи, які повинні бути покриті при проходженні тестів, визначаються на основі структурних критеріїв тестування C0, C1, C2. До них відносяться вершини, дуги, шляхи УГП, умови, комбінації умов тощо.

Принцип 2. На основі аналізу потоку даних, коли елементи, які повинні бути покриті, визначаються за допомогою потоку даних, інакше кажучи, інформаційного графа програми.

Тестування програми P по деякому критерію C означає покриття множини компонентів цієї програми $M = \{m_1, ..., m_k\}$ за елементами або по зв'язках, де $T = \{t_1, ..., t_k\}$ – кортеж не надлишкових тестів t_i [8].

Тест t_i не є надлишковим, якщо існує покритий їм компонент m_i з $M(P,C)$, не покритий жодним з попередніх тестів $t_1, ..., t_{i-1}$. Кожному t_i відповідає не надлишковий шлях p_i – послідовність вершин від входу до виходу.

Введемо в розгляд такі позначення.

$V(P,C)$ – складність тестування P за критерієм C – вимірюється максимальним числом не надлишкових тестів, що покривають всі елементи множини $M(P,C)$.

$DV(P,C,T)$ – залишкова складність тестування P за критерієм C – вимірюється максимальним числом не надлишкових тестів, що покривають елементи множини $M(P,C)$, що залишилися непокритими, після прогону набору тестів T. Величина DV строго і монотонно убуває від V до 0.

$TV(P,C,T) = (V - DV)/V$ – оцінка ступеня тестованості P за критерієм C.

Критерій закінчення тестування $TV(P,C,T) \geq L$, де $(0 \leq L \leq 1)$, L – рівень відтестованості, заданий у вимогах до програмного продукту.

Крок 4. ***Побудова УГП, що інтегрує модулі в єдину ієрархічну модель проекту.***

Побудова УГП відповідає статичному аналізу програми, завдання якого полягає в отриманні графа програми і залежних від нього і від критерію тестування безлічі елементів, які необхідно покрити тестами.

УГП компонента G, представлений у вигляді ієрархічної моделі (рис. 2). В ієрархічному УГП G входять до його складу компоненти представлені посиланнями на свої УГП G1 і G2 (рис. 3).

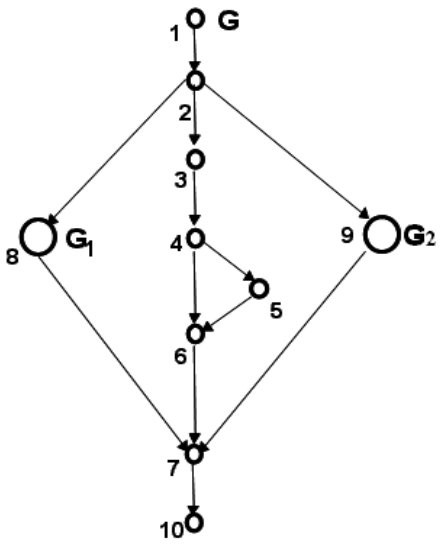

Рис. 2. Ієрархічна модель УГП компонента G

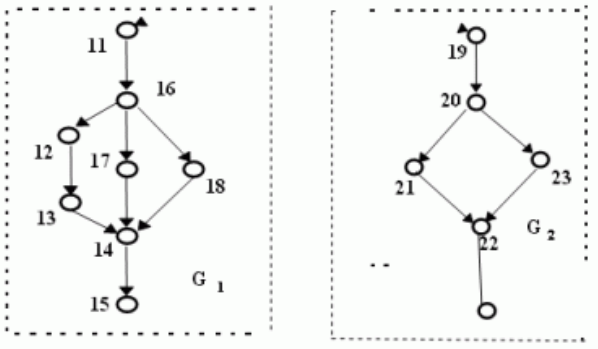

Рис. 3. Ієрархічна модель:
УГП компонент G1 і G2

Крок 5. ***Вибір тестових шляхів для проведення інтеграційного або системного тестування.***

Виділяють три підходи до побудови тестових шляхів:

– статичні методи;

– динамічні методи;

– методи реалізованих шляхів.

Статичні методи. Найпростіше і найбільш легко реалізоване рішення – побудова кожного шляху за допомогою поступового його подовження за рахунок додавання дуг, поки не буде досягнута вихідна вершина управляючого графа програми. Ця ідея може бути посилена в так званих адаптивних методах, які кожен раз додають тільки один тестовий шлях (вхідний тест), використовуючи попередні шляхи (тести) як керівництво для вибору подальших шляхів відповідно до деякої стратегії. Найчастіше адаптивні стратегії застосовуються по відношенню до критерію С1. Основний недолік статичних методів полягає в тому, що не враховується можлива нереалізуємість побудованих шляхів тестування [8].

Динамічні методи. Такі методи передбачають побудову повної системи тестів, що задовольняють заданому критерію, шляхом одночасного розв'язання завдання побудови, що покриває множину шляхів і тестових даних. При цьому можна автоматично враховувати можливість і неможливість реалізації раніше розглянутих шляхів або їх частин. Основною ідеєю динамічних методів є приєднання до початково реалізованих відрізків шляхів подальших їх частин так, щоб:

а) не втрачати при цьому реалізованості знов отриманих шляхів;

б) покрити необхідні елементи структури програми.

Методи реалізованих шляхів. Дана методика полягає у виділенні з множини шляхів підмножини всіх реалізованих шляхів. Після чого покриття множини шляхів будується з отриманої підмножини реалізованих шляхів.

Перевага статичних методів полягає в порівняно невеликій кількості необхідних ресурсів, як при використанні, так і при розробці. Проте їх реалізація

може містити непередбачуваний відсоток браку (нереалізованих шляхів). Крім того, у цих системах перехід від покриваючої множини шляхів до повної системи тестів користувач повинен здійснити вручну, а ця робота досить трудомістка. Динамічні методи вимагають значно більших ресурсів як при розробці, так і при експлуатації, однак збільшення витрат відбувається, в основному, за рахунок розробки та експлуатації апарату визначення реалізованості шляху (символічний інтерпретатор, вирішувач нерівностей). Перевага цих методів полягає в тому, що їхня продукція має деякий якісний рівень – реалізацію шляхів. Методи реалізованих шляхів дають найкращий результат.

Для вичерпного тестування ієрархічної моделі компонента G відповідно до критерію шляхів прогоняється свій відповідний набір трас. Цей набір трас достатній за умови, що G1 і G2 , в свою чергу, вичерпно протестовані. Щоб забезпечити виконання цієї умови відповідно до критерію шляхів, прогоняються всі траси.

Крок 6. ***Генерація тестів, що покривають тестові шляхи кроку 5.***

На цій фазі за відомими шляхами тестування здійснюється пошук відповідних тестів, що реалізують проходження цих шляхів.

Крок 7. ***Інтегральна оцінка тестованості проекту з урахуванням оцінок тестованості модулів-компонентів.***

Оцінка ступеня тестованості плоскої моделі визначається часткою прогнаних трас із набору необхідних для покриття відповідно до критерію C

$$TV(G,C) = \frac{V - DV}{V} = \frac{\sum PT_i(G)}{\sum P_i(G)}, \qquad (1)$$

де $PT_i(G)$ – тестовий шлях (t_i) в графі G плоскої моделі дорівнює 1, якщо він протестований (прогнаний), або 0, якщо ні.

Оцінка тестованості ієрархічної моделі визначається на основі врахування оцінок тестованості компонентів.

Якщо траса деякого тесту t_i УГП G включає вузли, що представляють компоненти $G_{j1},...,G_{jm}$, оцінка TV ступеня тестованості яких відома, то оцінка тестованості $PT_i(G)$ при реалізації цієї траси визначається не 1, а мінімальною з оцінок TV для компонентів. Інтегральна оцінка визначається співвідношенням:

$$TV(G,C) = \frac{V - DV}{V} = \frac{\sum PT_i(G) \cdot \sum (TV(G_{ij},C))}{\sum P_i(G)}, \quad (2)$$

де $PT_i(G)$ – тестовий шлях (t_i) в графі G дорівнює 1, якщо протестований, або 0, якщо ні.

У шлях PT_i графа G може входити j вузлів модулів G_{ij} зі своєю ступенем тестованості $TV(G_{ij},C)$ з яких ми беремо мінімум, що дає гіршу оцінку ступеня тестування шляху.

Крок 8. ***Повторення кроків 5 – 7 до досягнення заданого рівня відтестованості*** L**.**

Запропонований структурний алгоритм тестування спеціального програмного забезпечення полягає в застосуванні викладеної методики інтегральної оцінки тестування в циклі до досягнення заданого рівня відтестованості L (адекватності функціонування програмного продукту та оцінювання результатів тестування протягом заданого терміну його експлуатації в тестовому режимі). При цьому:

1)структурні критерії C0, C1 і C2 застосовуються на стадії тестування програми розробниками з повноцінним проходження всіх циклів із урахуванням принципів системного тестування.

Системне тестування розглядає систему, що тестується, в цілому і оперує на рівні користувача інтерфейсів. Основне завдання системного тестування – виявлення дефектів, пов'язаних із роботою системи в цілому, таких як невірне використання ресурсів системи, непередбачені комбінації даних рівня користувача, несумісність із оточенням, непередбачені сценарії використання, відсутня або невірна функціональність, незручність у застосуванні тощо. Системне тестування проводиться над проектом в цілому за допомогою методу «чорного ящика». Структура програми не має ніякого значення, для перевірки доступні лише входи і виходи, які бачить користувач. Категорії тестів системного тестування: повнота вирішення функціональних завдань, стресове тестування – на граничні обсяги навантаження вхідного потоку, коректність використання ресурсів (витік пам'яті, повернення ресурсів), оцінка продуктивності, ефективність захисту від перекручування даних і некоректних дій, перевірка інсталяції та конфігурації на різних платформах, коректність документації.

В нашому випадку, для спеціального програмного забезпечення, системне тестування доцільно проводити двічі – до і після інтегральної оцінки тестування програмного продукту. Перший етап проводиться з метою виявлення можливих помилок узгодження інтерфейсів, їх функцій і модулів системи без урахування особливостей (помилок) експлуатації, а другий – із урахуванням цих особливостей;

2) функціональні критерії застосовуються після структурних на стадії тестування програмного продукту розробниками (термін – тиждень), а також при експлуатації в режимі користувача при складанні акта тестування, (термін – тиждень, режим – щоденне щогодинне тестування однією групою користувачів);

3) після цього стохастичні критерії тестування застосовуються на стадії тестування програмного

продукту при його експлуатації в тому ж режимі з метою реалізації одного з трьох ключових принципів тестування програмного забезпечення – тестування на випадкових даних для великих обсягів детермінованих тестів.

Висновки

Таким чином, із метою оцінки якості спеціального програмного забезпечення та можливості його введення до експлуатації, на основі проведеного аналізу основних існуючих критеріїв та методів тестування програмних продуктів розроблені методика інтегральної оцінки ступеня тестованості спеціального програмного забезпечення та структурний алгоритм його тестування. Розроблений алгоритм циклічно повторює свої дії до досягнення заданого рівня відтестованості, причому структурні критерії застосовуються на стадії тестування програмного продукту розробниками з повноцінним проходженням всіх циклів із урахуванням принципів системного тестування; функціональні критерії – після структурних на стадії тестування розробниками та при експлуатації в режимі користувача при складанні акта тестування; стохастичні – на стадії тестування програмного продукту при його експлуатації в тому ж режимі з метою реалізації тестування на випадкових даних.

Одним із можливих напрямків подальших досліджень є вдосконалення запропонованого алгоритму за рахунок розширення множини критеріїв, що застосовуються для оцінки тестованості спеціального програмного забезпечення.

Список літератури

4. Біла книга 2012. ЗС України. – К.: Видання МОУ, 2013. – 78 с.

5. ДСТУ ISO 9004-2001. Системи управління якістю. Настанови щодо поліпшення діяльністю (ISO 9004:2000, IDT). – К.: Держстандарт України, 2001. – 44 с.

6. Шарапов О.Д. Системний аналіз: навч.-метод. посібник для самост. вивч. дисц. / О.Д. Шарапов. – К.: КНЕУ, 2003. – 154 с

7. Моисеев В.Б. Информационный подход к выбору решений в системах адаптивного тестирования / В.Б. Моисеев, Л.Г. Пятирублевый, К.Р. Таранцева // Матлы конф. «Анализ качества образования и тестирование». 22.03.2005. – М.: МО РФ, МЭСИ.

8. Тесты, тесты, тесты / под ред. М. Мацковского. – М.: Молодая гвардия, 1990. – 192 с.

9. Котляров В.П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова. – М.: БИНОМ, 2006. – 285 с.

10. Лямец В.И. Методы статистического анализа / В.И. Лямец. – Х.: ХВВКИУРВ, 1988. – 227 с.

11.Переверзев В.Ю. Технология разработки тестовых заданий: справоч. рук-во / В.Ю. Переверзев. – М.: E-Media, 2005. – 256 с.

Надійшла до редколегії 22.02.2013

Рецензент: д-р техн. наук, проф. Ю.В. Стасєв, Харківський університет Повітряних Сил ім. І. Кожедуба, Харків.

МЕТОДИКА ИНТЕГРАЛЬНОЙ ОЦЕНКИ СТЕПЕНИ ТЕСТИРОВАННОСТИ И СТРУКТУРНЫЙ АЛГОРИТМ ТЕСТИРОВАНИЯ СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

И.А. Романенко, К.С. Смеляков, И.В. Рубан, С.В. Алексеев, В.В. Калачева, А.П. Бабенко

Оценивание уровня подготовленности органов военного управления, частей и подразделов, личного состава Вооруженных Сил Украины в большинстве осуществляется неавтоматизированным способом, потому актуальной задачей настоящего является усовершенствование процедуры оценивания, обеспечения своевременности контрольных мероприятий, достижения максимальной объективности, повышения экономической эффективности за счет автоматизации процесса подготовки. При этом, одним из ключевых вопросов становится тестирование специального разрабатываемого программного обеспечения. Предложенные методика интегральной оценки степени тестованости специального программного обеспечения и структурный алгоритм его тестирования гарантируют достижение высокого уровня качества программного продукта.

Ключевые слова: *специальное программное обеспечение, управляющий граф программы, степень тестованости, набор трасс, критерий путей.*

METHOD OF INTEGRAL ESTIMATION OF DEGREE OF TESTING AND STRUCTURAL ALGORITHM OF TESTING OF THE SPECIAL SOFTWARE

I.O. Romanenko, K.S. Smelyakov, I.V. Ruban, S.V. Alekseev, V.V. Kalacheva, O.P. Babenko

Evaluation of level of preparedness of organs of military management, parts and subsections, personnel of Military Powers Ukrainy in majority is carried out by the manned method, that is why the relevant task of the present is an improvement of evaluation procedure, providing of timeliness of control measures, achievement of maximal objectivity, increase of economic efficiency due to automation of process of preparation. Thus, one of key questions is become by testing of the special developed software. Offered method of integral estimation of degree of testing of the special software and structural algorithm of his testing guarantee achievement high level of quality of software product.

Keywords: *special software, managing count of the program, degree of testing, set of routes, criterion of ways.*