МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 21.00.00.000 ПЗ

Група ІПм-22-2

Бурин Олесь

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Бурин Олесь Юрійович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та технології оцінки якості програмних рішень для автоматизованих систем

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Бурин О.Ю.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Тимків Дмитро Федорович, д.т.н., професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

доц.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Бурину Олесю Юрійовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та технології оцінки якості програмних рішень для автоматизованих систем ”

керівник проекту (роботи) Тимків Дмитро Федорович, д.т.н., професор

затверджені наказом закладу вищої освіти від “ _____ ” листопада 2023 р. № _____

2. Строк подання студентом проекту (роботи) 15 січня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування систем оцінки якості

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз систем забезпечення якості та сертифікації програмного забезпечення
2. Побудова функціональних схем та опис моделі якості програмного забезпечення
3. Опис технології оцінювання характеристик якості програмного забезпечення
4. Реалізація методів забезпечення якості та сертифікації програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Еволюція підходів до забезпечення якості програмного забезпечення (рис. 1.1)
2. Модель якості програмного забезпечення, що запропонована в стандарті ISO/IEC (рис. 1.2)
3. Функціональна схема сертифікації програмного забезпечення (рис. 1.3)
4. Структурна схема побудови сертифікаційної моделі якості (рис. 2.1)
5. Класифікація показників якості програмного забезпечення (рис. 2.2)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц., к.т.н. Вовк Р.Б.	
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____ _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	01.10.2023	виконано
2	Аналіз концепцій та алгоритмів предметної області	25.10.2023	виконано
3	Аналіз систем забезпечення якості та сертифікації програмного забезпечення	10.11.2023	виконано
4	Побудова функціональних схем та опис моделі якості програмного забезпечення	22.11.2023	виконано
5	Опис технології оцінювання характеристик якості програмного забезпечення	01.12.2023	виконано
6	Реалізація функціональності запропонованої інформаційної технології	15.12.2023	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	16.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 95 с., 27 рис., 4 табл., 44 джерела.

Тема: Моделі, методи та технології оцінки якості програмних рішень для автоматизованих систем.

Об'єкт дослідження: якість програмного забезпечення автоматизованих систем.

Мета роботи: дослідження науково обґрунтованої моделі якості програмного забезпечення системи автоматизованого контролю і технології оцінки її характеристик.

Предмет дослідження: методи побудови моделі та технології оцінювання характеристик якості програмного забезпечення системи автоматизованого контролю.

Результати дослідження:

Розроблено методики побудови сертифікаційної моделі програмного забезпечення системи керування, суть якої зводиться до відображення вимог замовника і галузевих стандартів на сукупність параметрів моделі, що складаються із множин вибраних характеристик та атрибутів якості, критеріїв відповідності та метрик, які обираються відповідно до рекомендацій стандартів.

Висновок:

В результаті досліджень отримано адекватну вимогам до програмного забезпечення модель та алгоритмічне забезпечення для проведення тестів визначення якості та забезпечено розрахунок фактичних показників якості для характеристик, які увійшли до моделі.

ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, СЕРТИФІКАЦІЯ, ТЕСТОВИЙ НАБІР ДАНИХ, АВТОМАТИЗОВАНА СИСТЕМА, ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ.

ABSTRACT

Master Thesis: 95 pp., 27 fig., 4 tab., 44 sources.

Thesis Subject: Models, methods and technologies for evaluating the quality of software solutions for automated systems.

Object of research: software quality of automated systems.

Research goal: research of scientifically substantiated model of software quality of automated control system and technology of evaluation of its characteristics.

Subject of research: methods of model construction and technology of evaluation of quality characteristics of software of automated control system.

The results:

Methods of building a certification model of management software, the essence of which is to reflect the requirements of the customer and industry standards on a set of model parameters consisting of sets of selected characteristics and quality attributes, compliance criteria and metrics selected according to the recommendations of the standards.

Conclusion:

As a result of the research, an adequate model for software requirements and algorithmic support for quality determination tests were obtained, and the calculation of actual quality indicators for the characteristics included in the model was provided.

QUALITY OF SOFTWARE, CERTIFICATION, TEST DATA SET, AUTOMATED SYSTEM, SOFTWARE DESIGN.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АП – аналоговий параметр

АСК – автоматизована система контролю

АСКП – автоматизована система контролю польотів

ЛА – літальний апарат

НТД – нормативно-технічна документація

ООП – об'єктно-орієнтоване проектування

ПЗ – програмне забезпечення

ПЗ АСК – програмне забезпечення автоматизованих систем контролю

ПС – програмна система

ТНД – тестовий набір даних

ВСТУП

Актуальність теми.

Якість програмного забезпечення — характеристика програмного забезпечення, ступінь відповідності програмного забезпечення до вимог. При цьому вимоги можуть трактуватись по-різному, що породжує декілька незалежних визначень терміну. Якість програмного забезпечення – набір властивостей продукту (сервісу або програм), що характеризують його здатність задовольнити встановлені або передбачувані потреби замовника. Поняття якості має різні інтерпретації залежно від конкретної програмної системи і вимог до неї.

Якість коду може визначатись різними критеріями. Деякі з них мають значення тільки з точки зору людини. Наприклад, форматування тексту програми — неважливо для комп'ютеру, але може мати велике значення для супроводу. Багато з існуючих стандартів кодування, що визначають специфічні для мови програмування угоди та задають низку правил, мають на меті полегшити супровід програмного забезпечення в майбутньому. Також існують інші критерії, що визначають чи «гарно» написаний код, наприклад, такі, як структурованість — ступінь логічного розділення коду на блоки.

Для забезпечення необхідного рівня експлуатаційних характеристик програмного забезпечення автоматизованих систем керування проводиться атестація та сертифікація. Однак, у зв'язку з відсутністю типових математичних моделей та формалізованих алгоритмів оцінки характеристик якості, а також засобів автоматизації випробувань, вартість цих процедур дуже висока. Через це сертифікаційні випробування програмного забезпечення автоматизованих систем керування проводяться не в повному обсязі, часто без дотримання процедури й рекомендацій сучасних стандартів якості програмного забезпечення, а сертифікація зводиться, як правило, лише до перевірки функціональної повноти програмного забезпечення та зручності його використання.

Повсякчас актуальною є проблема підвищення безпеки функціонування об'єктів, контрольованих за допомогою АСК. Одним із шляхів досягнення цієї мети є забезпечення необхідного рівня якості ПЗ АСК, що особливо важливо при експлуатації систем даного класу, оскільки вони широко використовуються в різних галузях і є критичними, бо пов'язані з безпекою життєдіяльності.

Наслідки низької якості розглянутих систем можуть бути досить серйозні і не раз призводили до аварій і катастроф. У зв'язку з цим, до характеристик ПЗ АСК висуваються особливо жорсткі вимоги, які задаються у відповідних галузевих стандартах та інших нормативних документах.

При цьому лишаються поза увагою такі важливі властивості ПЗ АСК, як точність, надійність, вірогідність допускового контролю та ін. Таке положення викликане тим, що для проведення сертифікації в повному обсязі потрібно в кожному конкретному випадку здійснювати спеціальні дослідження з метою розробки методів та алгоритмів обчислення показників характеристик, що пов'язано з виконанням додаткових обсягів робіт і витратами значних технологічних ресурсів.

Наприклад, ПЗ АСКП є критичним, бо призначено як для контролю діяльності екіпажів у польоті, так і для контролю двигунів та радіоелектронних комплексів. Однак з аналізу нормативних документів, по яким проводяться сертифікаційні випробування цих програмних комплексів, можна зробити висновок, що для більшості характеристик відсутні атрибути й метрики, а тому модель сертифікації не є повною та еквівалентною вимогам. Проте ці програмні комплекси використовуються авіакомпаніями, що при відсутності гарантованої вірогідності результатів контролю знижує безпеку польотів і може привести до інцидентів.

Зважаючи на це, актуальною задачею, яка раніше не досліджувалась, є дослідження методів побудови сертифікаційної моделі ПЗ АСК та технології оцінювання її характеристик, що необхідно для оцінки ступеня досягнення відповідності вимогам по кожній властивості ПЗ.

Мета досліджень полягає в дослідженні науково обґрунтованої моделі якості програмного забезпечення системи автоматизованого контроль і технології оцінки її характеристик.

Основні задачі, що підлягали вирішенню:

- опис концепції побудови сертифікаційної моделі якості ПЗ АСК із врахуванням рекомендацій стандартів якості ПЗ;
- дослідження методів та ефективних стратегій тестування ПЗ АСК із метою обчислення фактичних значень показників якості;
- розробка алгоритмів та ПЗ оцінки показників вірогідності виявлення подій контролю програмним забезпеченням, що сертифікується.

Об'єктом дослідження є якість програмного забезпечення автоматизованих систем.

Предметом дослідження – методи побудови моделі та технології оцінювання характеристик якості програмного забезпечення автоматизованих систем.

Методи досліджень.

В дослідженні використано методи аналізу вимог до ПЗ і стандартів якості ПЗ, методи інженерії якості програмних систем, порівняльний аналіз методів тестування, методи класифікації алгоритмів контролю, методи математичної логіки і теорії скінченних автоматів, апарат теорії ймовірностей та математичної статистики, методи аналізу і проектування ПЗ.

Наукова новизна отриманих результатів полягає в розробці методики побудови сертифікаційної моделі програмного забезпечення системи керування, суть якої зводиться до відображення вимог замовника і галузевих стандартів на сукупність параметрів моделі, що складаються із множин вибраних характеристик та атрибутів якості, критеріїв відповідності та метрик, які обираються відповідно до рекомендацій стандартів.

Практичне значення одержаних результатів полягає в тому, що отримано адекватну вимогам до програмного забезпечення модель та алгоритмічне забезпечення для проведення тестів визначення якості та

забезпечено розрахунок фактичних показників якості для характеристик, які увійшли до моделі.

Обсяг і структура роботи. Магістерська робота складається із вступу, трьох розділів, висновків, списку використаних джерел (44 найменування) і 1 додатку. Робота викладена на 95 сторінках друкованого тексту і містить 4 таблиці і 27 рисунків.

РОЗДІЛ 1.
АНАЛІЗ ПРОЦЕСІВ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА СЕРТИФІКАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

Одною з важливих та актуальних задач інженерії програмних систем є забезпечення необхідного рівня їх якості. Це особливо важливо для критичних систем, до яких належать автоматизовані системи керування, що широко використовуються на транспорті, в енергетиці, для моніторингу довкілля та в інших галузях.

Методів сертифікації якості програмного забезпечення стає все більше. Популярні підходи, основані на процесах, такі як ISO 9000 та SEI-CMM, змушують розробників програмного забезпечення жорстко притримуватись вибраних стандартів та процесів розробки.

Програмне забезпечення автоматизованих систем керування являє собою складні програмні комплекси, що придатні для використань у спеціалізованих обчислювальних системах і мають низку загальних істотних характеристик:

1) наявність загальних цілей і набору обов'язкових задач, що підлягають рішенню;
2) велика кількість елементів, що складають систему (програми, модулі і т.п.);
3) можливість виділення підсистем, які формуються з найбільш близьких по функціональних цілях груп елементів;
4) ієрархічна структура зв'язків між підсистемами;
5) наявність інтерактивних режимів роботи.

Програмне забезпечення автоматизованих систем керування складається із сотень модулів, що взаємодіють у процесі вирішення цільової задачі, якою є обробка параметричної інформації з метою прийняття діагностичних і управляючих рішень про стан об'єкта контролю та якість його функціонування, і має весь набір властивостей складних програмних систем. Для прийняття рішення з необхідною достовірністю потрібно, щоб програмна система мала високу надійність. Складність програмних комплексів і велике

число можливих маршрутів виконання усередині програмних модулів визначає, крім того, вимогу до стійкості системи стосовно помилок у вхідній інформації.

Такі системи відносяться до критичних систем цільового призначення, а тому перед допуском до експлуатації необхідно виконувати незалежний контроль рівня їх якості, тобто оцінювати множину властивостей програмного забезпечення шляхом сертифікаційних випробувань. Згідно стандартів термін сертифікація відповідності саме й означає дії третьої сторони (органа сертифікації та випробувальної лабораторії), спрямовані на підтвердження того, що програмне забезпечення відповідає встановленим вимогам стандартів та інших нормативних документів.

1.1 Аналіз проблем сертифікації програмного забезпечення та шляхи їх вирішення

Необхідно зазначити, що відсутність формалізованих моделей та методів роблять процедуру сертифікації трудомісткою й вартісною, внаслідок чого стримується широке її впровадження як у сучасні технології виготовлення програмного забезпечення, так і в процеси модернізації програмного забезпечення, а сертифікація часто проводиться не в повному обсязі та без належного обґрунтування отриманих результатів.

Таке положення викликано тим, що на сьогодні відсутні науково обґрунтовані методики розв'язування всього комплексу задач, пов'язаних із сертифікацією програмного забезпечення. Наукові дослідження в області якості програмних систем, в основному, присвячені питанням побудови систем забезпечення якості процесів життєвого циклу програмних систем та їх сертифікації у відповідності зі стандартами серії ISO9000 (CMM, TickIT та ін.). Прямо застосувати ці результати при сертифікації неможливо, бо вони не дозволяють оцінити досягнутий рівень якості програмних систем. Дійсна

робота присвячена дослідженню й розв'язанню цих проблем та розробці технології сертифікації програмного забезпечення.

Стандарт ISO/IEC 9126 регламентує зовнішні і внутрішні характеристики якості. Перші відображають вимоги до функціонування програмного продукту. Для кількісного встановлення критеріїв якості, за якими буде здійснюватися перевірка і підтвердження відповідності програмного забезпечення заданим вимогам, визначаються відповідні зовнішні вимірювані властивості (зовнішні атрибути) програмного забезпечення, метрики (наприклад, час виконання окремих компонентів), діапазони зміни значень і моделі їх оцінки. Метрики використовуються на стадії тестування або функціонування і називаються зовнішніми метриками. Вони являють собою моделі оцінки атрибутів. Внутрішні характеристики якості і внутрішні атрибути ПЗ використовуються для складання плану досягнення необхідних зовнішніх характеристик якості продукту. Для квантифікації внутрішніх характеристик якості застосовують внутрішні метрики, як інструмент перевірки відповідності проміжних продуктів внутрішнім вимогам до якості, які формулюються на процесах, що передують тестуванню.

Сертифікація програмного забезпечення – процедура перевірки відповідності характеристик програмного забезпечення вимогам, а сертифікаційні випробування призначені для експериментального визначення кількісних та якісних характеристик програмного забезпечення. Однак, вимоги до програмних систем носять суб'єктивний характер, оскільки вони формуються замовником, розробником і користувачем, а їх пряме використання при сертифікації може призвести до того, що одна й та сама характеристика програмних систем різними суб'єктами буде трактуватися по різному (змішування вимог), декілька характеристик можуть бути необґрунтовано об'єднані в одну (об'єднання вимог) та ін. Тому сертифікація, яка базується на таких вимогах, зазнає впливу суб'єктивних факторів, що може призвести до некоректних висновків щодо її результатів, які можуть бути як підтверджені, так і спростовані, або не визнані якоюсь із сторін.

1.2 Огляд сучасного стану оцінки якості програмного забезпечення

Проведемо огляд досліджень та основних отриманих результатів у цьому напрямку. На рис. 1.1 показано еволюцію підходів до оцінки якості та побудови моделей якості ПЗ. Основною проблемою є розробка конструктивних підходів до побудови базової моделі якості програмного забезпечення, котра б була прийнятною для різних класів програмного забезпечення і визнавалась розробником, замовником і користувачами.

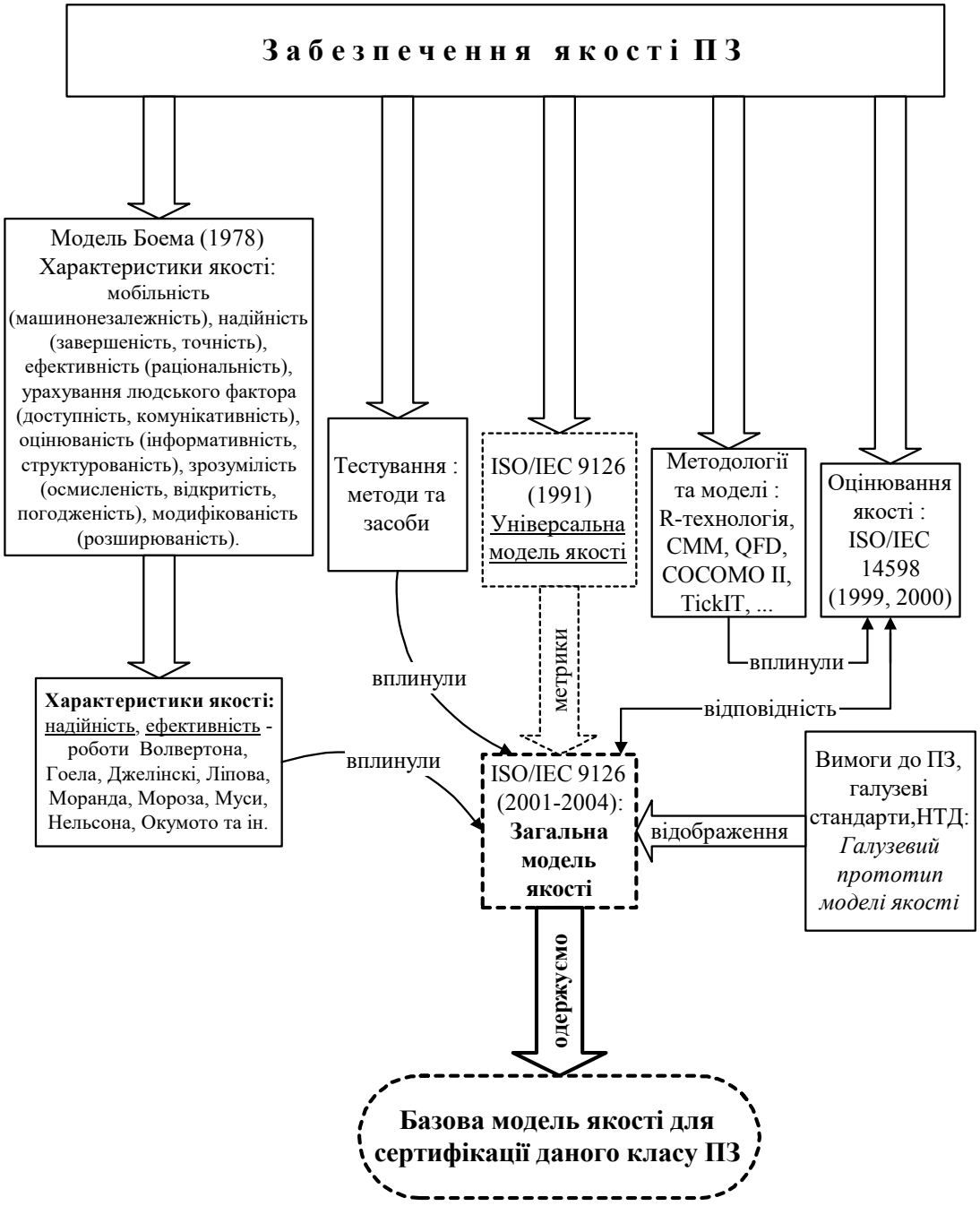

Рис. 1.1. Еволюція підходів до забезпечення якості програмного забезпечення

Існує найбільш повна на той час модель якості програмного забезпечення, яка була орієнтована переважно на розробника. Однак вона не отримала широкого впровадження, бо рівень інженерії програмного забезпечення того часу не дозволив ефективно застосувати модель при виготовленні програмного забезпечення. З визначених у цій моделі характеристик у подальших наукових працях основна увага була приділена надійності програмного забезпечення, бо для розрахунку фактичних значень цієї характеристики можна було ефективно застосувати тестування, а також тому, що тогочасне інструментальне середовище розробки та реалізації програмного забезпечення було недосконалим.

На практиці контроль якості обмежувався, в основному, застосуванням процедур динамічного тестування, що давало можливість лише виявити помилки на заданих ТНД. Перевірка ж відповідності вимогам нормованих показників або не проводилась взагалі, або це робилося формально. З виходом першої редакції ISO/IEC 9126, а також розробки моделей COCOMO II, CMM, QFD, TickIT та ін., почали проводитися дослідження по їх впровадженню у процес виготовлення ПС.

Згодом з'являється перероблена і розширена серія стандартів ISO/IEC 9126, яка приведена у відповідність із стандартами ISO/IEC 14598 (процеси оцінки програмного забезпечення), і містить рекомендації щодо створення загальної моделі якості та запровадження метрик для оцінки її характеристик. Саме ця модель узята за основу для побудови технології оцінювання якості програмного забезпечення при його сертифікації (рис.1.2).

В даний час програмне забезпечення сертифікується, як правило, акредитованими органами сертифікації за обраною схемою. У більшості випадків проводяться тестові випробування, аналізуються отримані результати й групою експертів приймається рішення про можливість видачі сертифіката відповідності програмного забезпечення вимогам.

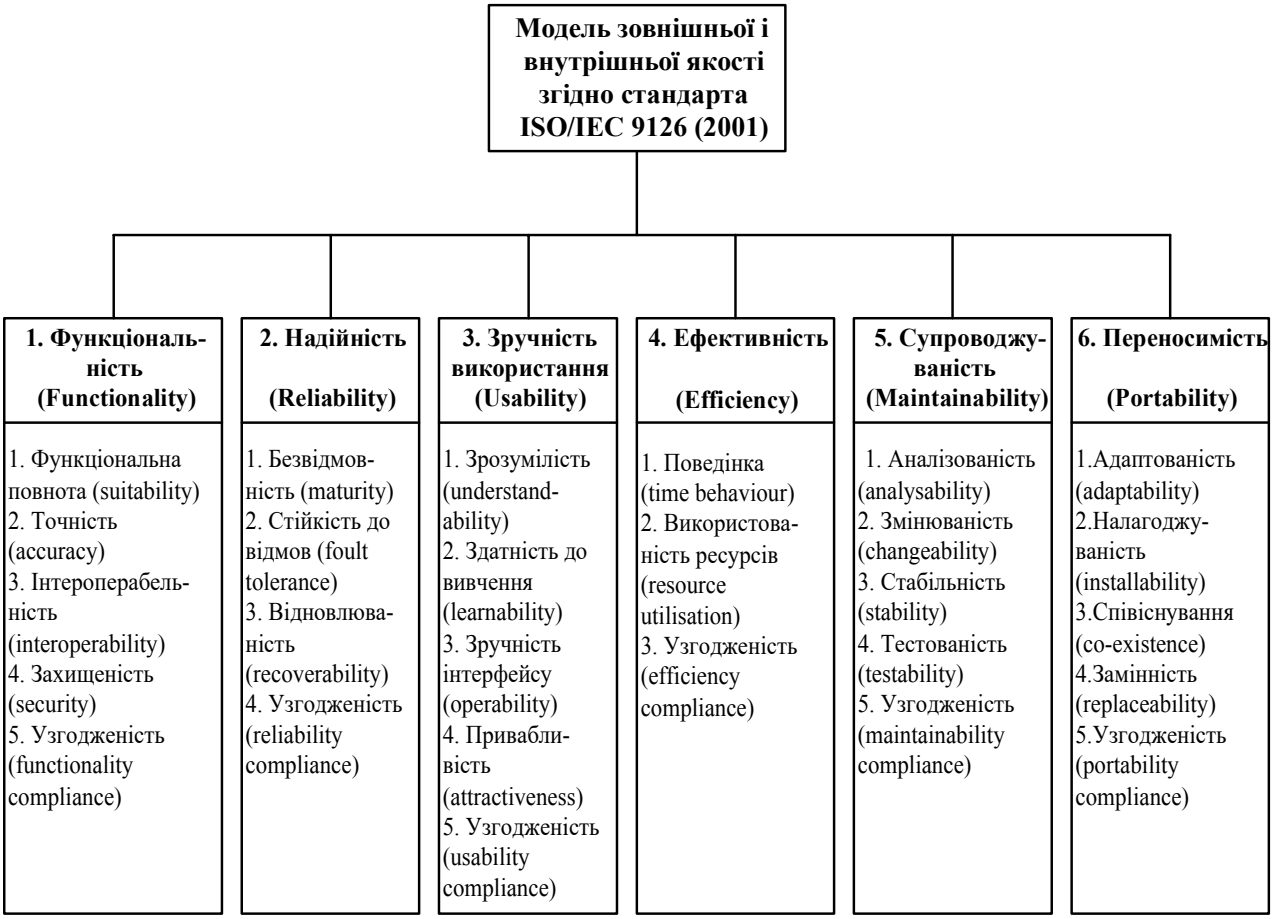

Рис. 1.2. Модель якості програмного забезпечення, що запропонована в стандарті ISO/IEC 9126

Однак необхідно відзначити, що програма сертифікаційних випробувань носить суб'єктивний характер і зводиться, в основному, до перевірки правильності функціонування програмного забезпечення на вузькому наборі тестів, зручності інтерфейсу користувача й достатності документації для експлуатації програмного забезпечення.

Формування вимог до кожного з класів програмного забезпечення проводиться спрощено, як правило, без узгодження з групами стандартів якості, без побудови моделі якості та без врахування повноти й достатності ТНД і обґрунтування вірогідності отриманих результатів. Дана робота присвячена побудові науково обґрунтованої технології сертифікації, яка базується на моделі якості, що розроблена у відповідності з рекомендаціями стандартів.

Передусім дамо більш детальний огляд сучасного стану проблем сертифікації, виходячи з рекомендацій стандартів і висвітлення цих проблем у науковій літературі. Необхідно відзначити, що вищевказані проблеми розглянуті в науковій літературі недостатньо. Першу і другу проблему в основному зводять до постулювання необхідності формування моделі якості. Для різних класів прикладного програмного забезпечення (зокрема, для класу програмного забезпечення АСК) ці питання не розглянуті (модель не побудована, питання формування набору належних характеристик і визначення фактичних показників якості не пророблені). Третю проблему, звичайно, зводять до автоматизації тестування, що значно вужче автоматизації процесу випробувань.

Стосовно першої проблеми відзначимо, що як випливає з рекомендацій стандартів, сертифікацію програмного забезпечення слід проводити із залученням третьої сторони (незалежних акредитованих лабораторій сертифікації), автоматизуючи процес випробувань впритул до етапу ухвалення рішення про сертифікацію відповідності, за яке цілком відповідає колектив, що проводив сертифікаційні випробування. Домінуючим підходом до побудови процедури оцінки програмного забезпечення при сертифікації поступово стає оцінка якості програмного забезпечення відповідно до його призначення.

Функціонально процес сертифікації програмного забезпечення буде складатися з наступних етапів :

1) Визначення сертифікаційних вимог до програмного забезпечення та побудова моделі якості програмного забезпечення.
2) Розробка алгоритмів та програмного забезпечення визначення фактичних показників якості.
3) Автоматизація процесу сертифікаційних випробувань.

Організація процесу сертифікації показана на рис.1.3.

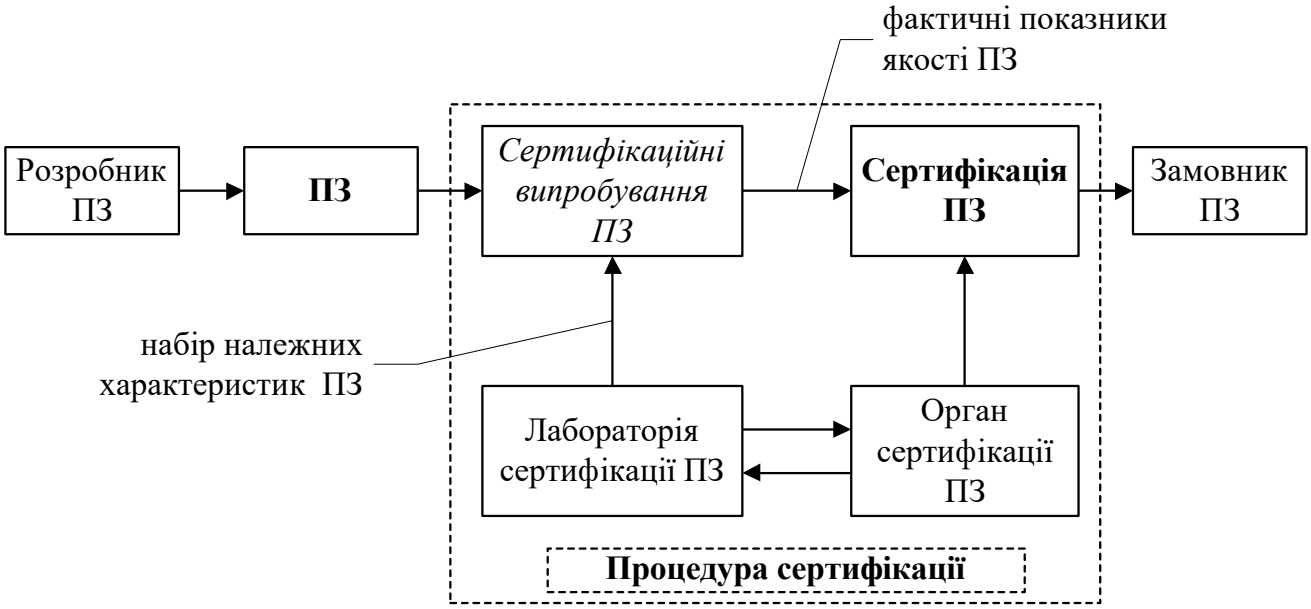

Рис. 1.3. Функціональна схема сертифікації програмного забезпечення

Група стандартів, що мають відношення до якості програмного забезпечення, складається з декількох серій. Стандарти серії ISO9000, а також моделі TickIT та CMM регламентують правила створення якісних програмних систем шляхом уведення власної системи керування якістю.

Оцінювати якість програмного забезпечення можна й у відповідності зі стандартами серії ISO/IEC14598, що пропонують способи оцінки характеристик продукту, запозичаючи, однак характеристики якості, визначення загальних вимог до програмного забезпечення, а також критерії його оцінки з ISO/IEC 9126.

Тому можна констатувати, що основною загальноприйнятою серією стандартів, яка висуває загальні вимоги до програмного забезпечення, є серія ISO/IEC 9126. Відповідно до нової класифікації, група стала називатися характеристикою, підгрупа - підхарактеристикою, показник - атрибутом підхарактеристики, елемент показника - елементом атрибута. Модель якості змінилася і у частині приналежності підхарактеристик, деякі з яких перейшли з однієї характеристики в іншу (наприклад, точність перейшла з характеристики надійність у функціональність). Крім того, деякі підхарактеристики визнані зайвими, бо мають лише непряме відношення до

якості програмного забезпечення (наприклад, документованість із usability). Останні в основному мають відношення до процесів життєвого циклу програмного забезпечення, або були виведені із загальної моделі якості, оскільки вони більше відносяться до процесів оцінки програмного продукту (стандарти серії ISO/IEC 14598 parts 1-6). Ряд підхарактеристик був доданий (наприклад, replaceability у переносимості), а деякі були замінені на аналогічні, чи близькі за змістом (наприклад, замість trainability введена learnability).

Таким чином, ISO/IEC 9126 був переглянутий, в основному, із метою узгодження його із серією ISO/IEC 14598, і випущена серія ISO/IEC 9126. Загальні характеристики якості програмного забезпечення, що розроблені в цій серії, є загальновизнаними на сьогоднішній день.

Часто будують модель на основі загальної моделі (ISO/IEC 9126). Модель якості, є загальною і не деталізована для застосування в якості моделі довільного класу прикладного програмного забезпечення. Доцільно використати цю модель для побудови моделей якості програмного забезпечення різної орієнтації. Для цього необхідно виділити належні даному класу програмного забезпечення характеристики якості з їх атрибутами, здійснити вибір метрик, а також множин вагових коефіцієнтів, тобто побудувати модель якості конкретного класу програмного забезпечення.

Розглядаючи другу проблему сертифікації, відзначимо, що кожний з показників якості визначається характеристикою, підхарактеристикою, набором атрибутів із метриками їхнього виміру і ваговими коефіцієнтами. Незалежно від типу метрики, для виміру фактичного значення більшості атрибутів характеристик якості необхідно проводити випробування, що являють собою в переважній більшості випадків прогін ТНД і аналіз результатів тестування.

У стандартах рекомендовані метрики для виміру підхарактеристик загальної моделі. Однак вони пропонують лише загальний підхід, залишаючись на рівні підхарактеристик, що не дає можливості розраховувати значення атрибутів чи елементів атрибутів якості. Відзначимо, що для

більшості характеристик не існує яких-небудь загальних аналітичних методів оцінки фактичних показників, що дозволяли б розраховувати значення атрибутів якості. Тому для виділення конкретних метрик, які складаються з методу і шкали виміру атрибута якості, необхідно для кожного класу програмного забезпечення, що розглядається, звертатися до відповідних галузевих стандартів і нормативних документів. При цьому вимоги галузевих стандартів треба співвідносити з характеристиками загальної моделі якості. Фактичні показники якості визначаються за допомогою тестування.

Стосовно третьої проблеми відзначимо, що державні стандарти рекомендують впроваджувати інструментальні обчислювальні засоби і спеціальне програмного забезпечення для забезпечення ефективних випробувань. Однак принципи, на які спирається структура такого програмного забезпечення, досліджені з точки зору проведення тестових випробувань, а не визначення показників якості. Вважається, що автоматизації підлягають процеси генерації ТНД та тестування, але тестування програмного забезпечення складає лише частину сертифікаційних випробувань. Крім того, необхідно автоматизувати процес формування моделі якості для програмного забезпечення даного класу (включаючи вибір метрик і вагових коефіцієнтів), генерації ТНД, визначення показників якості, оцінки результатів випробувань й ухвалення рішення, а також деякі інші.

Сертифікаційні випробування строго не регламентовані у загальних стандартах про випробування програмного забезпечення. Основні рекомендації стандартів полягають у складанні, узгодженні й виконанні плану та графіка випробувань, а також визначенні технічних і програмно-інструментальних засобів проведення випробувань.

У нашому випадку ці положення повинні бути погоджені між трьома сторонами:

(1) – розробник програмного забезпечення,

(2)– замовник програмного забезпечення (користувач),

(3) – орган сертифікації і лабораторія сертифікації.

В якості конкретного приклада програмного забезпечення АСК розглядається програмне забезпечення автоматизованих систем.

1.3 Проблеми сертифікації якості програмного забезпечення в автоматизованих системах

Відповідно стандарту наземні системи автоматизованої обробки польотної інформації повинні бути сертифіковані на відповідність вимогам державного стандарту України і нормативного документа.

Обробка інформації, записаної бортовими реєстраторами, є актуальною задачею, яка дозволяє забезпечувати безпеку польотів. За допомогою аналізу польотної інформації вирішуються задачі контролю режимів польоту й перевірка виконання правил льотної експлуатації апарату, оцінюється працездатність і визначаються причини авіаційних інцидентів.

У документах ІСАО високо оцінюється інформація БР і рекомендується впроваджувати обробку польотної інформації з метою запобігання авіаційних подій, вивчення дій екіпажа й поліпшення технічного обслуговування повітряних суден. На Україні використання польотної інформації всіма експлуатантами ЛА є обов'язковим і регламентується документами.

Програмне забезпечення АСКП призначено для обробки параметричної інформації, яка є складною структурованою інформацією, що містить параметри польоту даного об’єкту (висота, швидкість, положення рулів висоти, кут крену, кут тангажу, положення елеронів, кути відхилення ручок керування двигунами, обороти двигунів і багато ін.), розпізнавальні дані (час, дата, номер борта, номер рейса) та деяку контрольну інформацію. Копія польоту ЛА, зчитана зі стрічки бортового реєстратора, має складну організацію, що характеризується циклічним повторенням кадрів інформації, структурною організацією каналів реєстрації усередині кадру, наявністю розпізнавальних даних. ПІ відбиває динаміку руху й стан систем ЛА на протязі польоту. З метою контролю за діями екіпажа й системами ЛА, проводиться

післяпольотна обробка ПІ за допомогою спеціального програмного забезпечення контролю польотів.

Контроль польотів – це сукупність дій по обробці ПІ з метою перевірки виконання екіпажем норм льотної безпеки, оцінки якості пілотування і стану систем ЛА. У процесі контролю політ розбивається на ряд етапів: передзлітне рулювання, зліт, набір висоти, політ згідно з маршрутом, зниження, глісада, відхід на друге коло (якщо він мав місце), посадка, післяпосадкове рулювання. Дії екіпажа, режими роботи систем ЛА, ознаки етапів, готовності настання контрольованих подій та інші характеристики польоту регламентуються алгоритмами контролю. Алгоритми контролю польотів складаються розробниками даного типу ЛА, затверджуються Генеральним конструктором і являють собою складні й громіздкі логіко-алгебраїчні вирази, що містять десятки предикатів і логічних функцій, які регламентують поведінку ЛА як об'єкта контролю протягом усіх етапів польоту.

Розглянемо сучасний стан сертифікації програмного забезпечення контролю польотів. Дотепер, випробування проводяться із залученням колективу експертів, які, користуючись графіками та таблицями значень контрольованих параметрів, проводили підтвердження або спростування виявлення подій контролю програмними комплексами, що проходили випробування. Така процедура є трудомісткою і витратною, з високим впливом суб'єктивного фактора. Крім того, випробування проводились без дослідження достатності й повноти використовуваних ТНД, а через відсутність засобів автоматизації, обмежувались перевіркою лише на декількох тестових наборах даних. Цю проблему може вирішити впровадження сучасної науково обґрунтованої комп'ютеризованої технології сертифікації програмного забезпечення подібних технічних систем.

Однак, впровадження для випробувань програмного забезпечення АСКП засад технології сертифікаційних випробувань потребує деяких подальших досліджень. Необхідно ввести в розгляд призначення й спеціалізацію програмного забезпечення АСКП, обумовлену у вітчизняному стандарті і

нормативному документі Міждержавного Авіаційного Комітету країн СНД. Стандарт визначає набір задач, що підлягають рішенню, фіксує нормовані показники вірогідності настання подій контролю і декларує ряд інших вимог.

Передусім розглянемо склад програмного забезпечення АСКП і основні вирішувані ним задачі.

1) У копії польоту аналогові параметри записані у виді кодової величини. Для перетворення значень коду параметрів у фізичні величини служать градуювальні характеристики, котрі є індивідуальними для кожного датчика. Веденням бази даних градуювальних характеристик для ЛА, експлуатованих в авіапідприємстві, займається підсистема підготовки й ведення градуювальних характеристик. Така підсистема наявна у будь-якій системі обробки параметричної інформації.
2) Однією з головних є задача відтворення польотної інформації, яка полягає в графічному представленні зміни аналогових параметрів і разових команд у часі. Комплекс програм відтворення параметричної інформації присутній у будь-якій АСК. Він забезпечує автоматизовану обробку копії польоту з наступною видачею графічної й табличної інформації на зовнішні пристрої ЕОМ. Під автоматизованою обробкою копії польоту розуміється:
   - забезпечення можливості аналізу копії польоту при графічному чи табличному представленні АП і РК;
   - синхронна індикація розпізнавальних даних і часу;
   - можливість аналізу як копії у цілому, так і її фрагментів.
3) Основною проблемою, що встає перед програмним забезпеченням АСКП, є рішення задач допускового контролю. Відповідний комплекс програм повинний забезпечувати реалізацію всіх алгоритмів контролю, встановлених для даного типу ЛА. Цей комплекс характеризується ухваленням рішення про перебування об'єкта контролю в деякому стані. Для цього обчислюється складна логічна умова і визначається настання тієї чи іншої події контролю. Істотною проблемою для будь-якого

комплексу, що вирішує задачі контролю, є визначення вірогідності настання подій контролю й підтвердження факту перебування погрішності оцінки в межах допуску.

4) Задача контролю якості виконання польоту полягає в здійсненні контролю за виконанням екіпажами ЛА режимів і правил льотної експлуатації. Для рішення цієї задачі програмного забезпечення повинне мати наступні властивості:
   - повинні бути реалізовані алгоритми контролю в обсязі не меншому встановленого Генеральним конструктором ЛА;
   - наявність процедури підтвердження виявлених подій контролю;
   - наявність процедури формування інформаційного “портрета” польоту в табличному вигляді для візуальної оцінки показників якості пілотування.

На черзі тепер стоїть задача формалізації вимог до програмного забезпечення контролю польотів і побудова цільової моделі якості програмного забезпечення АСКП, придатної для сертифікації.

Висновки до розділу

В даному розділі на основі аналізу стандартів та сучасного стану сертифікації та контролю якості і виділено три основні проблеми, а саме: розробка конструктивних методів побудови сертифікаційної моделі якості ПЗ, розробка працездатних процедур розрахунку фактичних значень показників моделі та побудова засобів автоматизації сертифікаційних випробувань. Огляд стандартів та наукових публікацій дозволяє зробити висновок, що найбільш логічним та до розробки стандартної форми запису вимог та побудови сертифікаційної моделі є використання стандартів ISO/IEC.

РОЗДІЛ 2.
ПОБУДОВА ФУНКЦІОНАЛЬНИХ СХЕМ ТА ОПИС МОДЕЛІ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Першим етапом робіт по сертифікації програмного забезпечення є формулювання вимог у деякій стандартній, придатній для всіх сторін, формі. Такою формою може бути модель якості, що побудована у відповідності до стандарту. Другим етапом робіт є проведення випробувань програмного забезпечення для оцінки того факту, наскільки його характеристики задовольняють висунутим вимогам.

2.1 Концепція побудови сертифікаційної моделі якості програмного забезпечення

На основі аналізу існуючих формалізацій, що містяться у стандартах, було вибрано найбільш повний класифікатор, який запропонований у серії стандартів. Дійсно, більш логічним та економічним підходом до розробки стандартної форми запису вимог є використання класифікації характеристик програмного забезпечення, що викладена в стандарті якості ISO/IEC 9126-1. В цьому стандарті дана вичерпна класифікація характеристик якості програмного забезпечення, яка налічує багато підхарактеристик.

Однак, з іншого боку, стандарти визначають загальні поняття й рекомендації, які не можна прямо використати для конкретних обчислень рівня якості програмного забезпечення. Тому для кожного класу програмного забезпечення необхідно будувати модель якості з огляду на предметну область і специфіку використання програмного забезпечення. Відзначимо, що концепція аналітичної оцінки характеристик якості, яка базується на загальній моделі якості, не враховує специфіку різних класів програмного забезпечення і є в галузевому розумінні занадто широкою.

Тому для програмного забезпечення АСК пропонується відображати галузеві нормативні вимоги на характеристики загальної моделі, звужуючи її. У такий спосіб одержимо шукану модель сертифікації, у якій вагові множники при атрибутах якості будуть відповідати за специфіку предметної області. (Наприклад, для програмного забезпечення АСК функціональна повнота й точність важливіше мобільності).

Такий загальний підхід пропонується використати для створення концепції побудови моделі якості програмних систем, що функціонують у різних галузях. Основою концепції може служити принцип доцільної повноти моделі для кожного класу програмного забезпечення, що розглядається. Дійсно, різні класи програмного забезпечення мають різний пріоритетний набір розв'язуваних задач, який залежить від специфіки використання програмного забезпечення, що приводить нас до необхідності розробки концепції побудови моделі якості галузевого програмного забезпечення, яка придатна для використання при сертифікаційних випробуваннях.

Концепція визначає систему відображень для переходу від моделі вимог до стандартизованої моделі якості, яку можна використати для сертифікації програмного забезпечення. При побудові концепції слід врахувати вимоги замовника сертифікації, галузевих стандартів та нормативно-технічна документація, вимоги користувача,а також рекомендації державних стандартів, після чого відобразити отриману множину вимог на характеристики якості.

Концепція побудови сертифікаційної моделі якості програмного забезпечення АСК складається з наступних етапів :

1) Ґрунтуючись на сертифікаційних вимогах до програмного забезпечення АСК, будується нормативна модель вимог шляхом об'єднання функціональних і нефункціональних вимог та обмежень. Нормативна модель вимог, у першу чергу, залежить від набору вимог замовника сертифікації та галузевих вимог до програмного забезпечення. В цій моделі можуть з'являтися й конфліктні вимоги, що випливає із

суперечливості деяких вимог замовника і вимог галузевих та міжнародних стандартів.

2) Нормативна модель вимог не є конструктивною, тому що вимоги до програмного забезпечення АСК не стандартизовані (не задані атрибути та метрики для їх виміру). На основі аналізу стандартів, вибираємо найбільш повний класифікатор, що запропонований у стандартах ISO/IEC 9126, і будуємо загальну модель якості. Ґрунтуючись на базових програмних комплексах програмного забезпечення АСК, виконуємо поділ характеристик якості загальної моделі на критичні й другорядні.
3) Здійснюємо відображення вимог з нормативної моделі на загальну модель якості з вибіркою характеристик та сполученням атрибутів. Для вимог, які не мають відповідності у загальній моделі за назвою, шукаємо аналог за змістом. Конфліктні характеристики узгоджуємо. Сукупність вибраних елементів загальної й нормативної моделей складе сертифікаційну модель якості, у якій відповідність показників атрибутів вимогам залежить від обмежень.

Розглянемо зазначені етапи більш детально. На першому етапі пропонується з групи стандартів якості виділити базові показники якості, зокрема, для звуження множини показників, що перевіряються. Дійсно, якщо ввести до сертифікаційної моделі всі характеристики та підхарактеристики, що рекомендуються в стандарті якості, то ми одержимо необґрунтовано трудомістку й вартісну процедуру по їх перевірці.

Базові показники безпосередньо відбивають якість функціонування програмного забезпечення у зв'язку з його призначенням (найважливіші підхарактеристики – це функціональна повнота, безвідмовність, але не інтероперабельність, адже останню в разі потреби можна включити у функціональність програмного забезпечення). Іншими словами, базові показники якості характеризують ступінь виконання переліку функцій відповідно вимогам до програмного забезпечення. Базові показники знаходяться у взаємній відповідності з вимогами бо відбивають основну групу

вимог до програмного забезпечення, тому що для досліджуваного класу програмного забезпечення насамперед повинні бути перевірені всі належні функції у зв'язку з його призначенням.

Крім того, зауважимо, що до кожної характеристики якості додана важлива підхарактеристика - узгодженість, яка означає, що кожна характеристика повинна задовольняти рекомендаціям і вимогам стандартів для даного класу програмного забезпечення, тобто галузевим стандартам, чи нормативним документам на програмне забезпечення даного типу, яке функціонує в даній предметній області.

Критичні показники названі так за аналогією з назвою відповідних систем. Для програмного забезпечення АСК вони пов'язані з оцінкою функціонування об'єкта контролю і тому мають вирішальний вплив на якість програмного забезпечення критичних систем. Формування набору критичних показників здійснюється у відповідності з наступними етапами:

1) Відображення вимог галузевих стандартів і нормативних документів на множину уніфікованих показників якості загальних стандартів.
2) Доповнення отриманої множини обов'язковими характеристиками якості із загальних стандартів, що не ввійшли до відображення.
3) Вибір з отриманої моделі якості тих характеристик, що пов'язані з оцінкою функціонування об'єкта контролю.

Побудована процедура аж ніяк не рекомендує включення в модель якості всіх показників, установлених стандартом. Навпроти, експертам із лабораторії сертифікації рекомендується, враховуючи емпіричні критерії, розумно обмежити кількість характеристик і підхарактеристик якості при побудові моделі, не погіршуючи, однак, застосовності моделі й точності оцінки. Не слід, також, об'являти всі підхарактеристики критичними.

З метою спрощення моделі в дійсній роботі пропонується, по-перше, ввести в модель тільки критичні характеристики (функціональність, надійність і відображення необхідних галузевих характеристик), а, по-друге – потрібні другорядні характеристики. Побудована в такий спосіб модель забезпечить

належний рівень якості критичних систем цільового призначення, якими є АСК, а також достатність і повноту порівняння конкуруючих продуктів у даній галузі.

Застосуємо описаний метод для програмного забезпечення АСК (рис.1.2) :

1) Функціональність. Підхарактеристики функціональна повнота (1.1) і точність (1.2) є критичними, а інтероперабельність (1.3) – не критична, оскільки вона не впливає на оцінку функціонування об'єкта контролю. Захищеність можна опустити через те, що критичні системи працюють у сеансі з експертами, а не з користувачами. Зауважимо, що узгодженість функціональності з галузевими вимогами проводиться в обов'язковому порядку шляхом відображення показників. Перші дві підхарактеристики (suitability (1.1) та accuracy (1.2)) звичайно з'являються після відображення галузевих вимог і вимог користувача на множину показників стандарту ISO/IEC 9126 (2001).
2) Надійність. Безвідмовність (2.1) – критична підхарактеристика, а стійкість до відмов (2.2) і відновлюваність (2.3) можуть бути як критичними, так і ні. Це залежить від того, де використовується АСК: в режимі реального часу, чи після функціонування об'єкта. У другому випадку останні дві підхарактеристики навіть не обов'язково вводити до моделі якості.
3) Зручність використання. Здатність до вивчення (3.2) і зручність інтерфейсу (3.3) по великому рахунку є другорядними підхарактеристиками (хоча інколи їх можна переводити до критичних, особливо (3.3)). Узгодженість зручності використання (3.5) являється критичною підхарактеристикою, бо потребує погодженості з цільовими вимогами користувача до програмного забезпечення.
4) Не вводимо до моделі якості характеристику ефективність (efficiency), як таку, що не впливає на здатність системи до контролю.
5) Супроводжуваність. Змінюваність (5.2) – критична підхарактеристика, бо змінюваність коду не повинна впливати на якість функціонування програмного забезпечення у цілому.

6) Переносимість. Адаптованість (6.1) і налагоджуваність (6.2) є другорядними підхарактеристиками, які теж уведемо до моделі, а на інші можна не зважати.

Таким чином, на основі проведеного аналізу до базової моделі якості програмного забезпечення АСК вибрано п’ять характеристик із виділеними підхарактеристиками.

Врешті, із метою деталізації моделі необхідно вирішити задачу визначення атрибутів підхарактеристик. Атрибути й елементи атрибутів пропонується виділяти з галузевих стандартів і нормативних документів, а для підхарактеристик що залишилися – із загальних стандартів якості, оскільки галузеві документи задають базові вимоги до програмного забезпечення, що функціонує в заданій предметній області. Якщо галузеві нормативні документи відсутні, то вибір атрибутів варто залишити на розсуд експертів із лабораторії сертифікації, які повинні керуватися загальними стандартами якості програмного забезпечення, а також вимогами технічного завдання на розробку програмного забезпечення.

Зауважимо, що з точки зору користувача вимоги до властивостей програмного забезпечення виділяються з галузевих стандартів і нормативних документів, а з точки зору органів, що перевіряють (орган сертифікації), і сторони, що проводить сертифікаційні випробування (акредитована лабораторія сертифікації), ці показники повинні бути уніфікованими і вибиратися із загальних стандартів якості програмного забезпечення. Нарешті, розробник програмного забезпечення може користатися власною системою оцінки якості, що може бути заснована на контролі ряду вищезгаданих показників, а також деяких внутрішніх характеристик якості програмного забезпечення. З вищесказаного випливає, що на черзі тепер стоїть задача відображення вимог галузевих стандартів на множину показників загальних стандартів і визначення нормативних показників.

Відображення множини вимог галузевих стандартів і нормативних документів на множину показників загальних стандартів з якості необхідно проводити по трьох основних причинах :

1) Для сертифікаційних випробувань необхідно використовувати уніфіковані показники якості, зафіксовані в загальних стандартах на програмне забезпечення.
2) Фахівцям із лабораторії сертифікації, що проводять випробування, не зовсім зрозумілі вимоги галузевих стандартів і нормативних документів до програмного забезпечення, що сформульовані в термінах предметної області.
3) Система метрик характеристик якості пророблена в загальних стандартах, а обмеження та інструкції по обчисленню кожного з атрибутів задані в галузевих стандартах і нормативних документах.

Відображення вимог галузевих нормативних документів і показників внутрішньої системи оцінки якості програмного забезпечення на показники загальних стандартів є не чим іншим, як співвіднесенням вимог користувача й вимог розробника на множині показників якості загальних стандартів. Необхідно знайти узгодження множин вимог користувачів і розробників на множині уніфікованих показників загальних стандартів якості, а для тих показників, що не потрапили в результуючу множину, можна використати деякі відомі методи узгодження, наприклад метод SQFD.

Подібний підхід (відображення вимог нормативних документів на множину уніфікованих характеристик якості загальних стандартів) можна застосувати для будь-яких класів програмного забезпечення.

Діаграма побудови сертифікаційної моделі якості програмного забезпечення АСК приведена на рис.2.1.

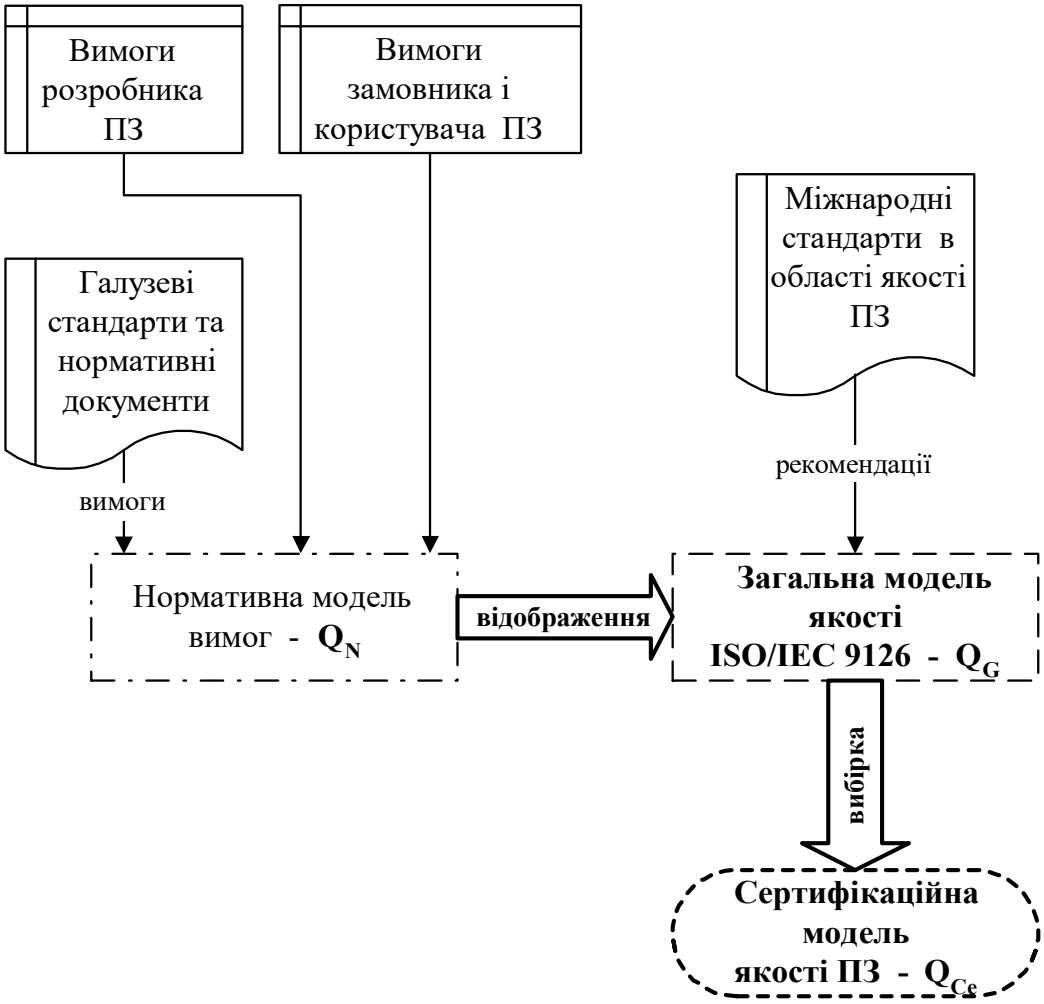

Рис. 2.1. Структурна схема побудови сертифікаційної моделі якості програмного забезпечення системи автомати

2.2 Принципи оцінки рівня якості програмного забезпечення

Розглянемо процедуру побудови сертифікаційної моделі якості програмного забезпечення АСК, котра б враховувала, з одного боку, вимоги замовника програмного забезпечення та галузевих стандартів, а з іншого – максимально задовольняла рекомендаціям міжнародних та національних стандартів з якості програмного забезпечення. Модель якості насамперед буде складатися з показників якості, які пропонується класифікувати згідно з наявними базовими програмними комплексами цих систем, що показано на рис.2.2.

Рис. 2.2. Класифікація показників якості програмного забезпечення АСК

Обрані показники якості являються характеристичними, бо вони є типовими для будь-яких автоматизованих систем контролю динамічних і стаціонарних об'єктів за вимірювальною інформацією, оскільки ці системи обов'язково містять у собі вищенаведені комплекси й програми. Можливо, що з огляду на специфіку предметної області й класів розв'язуваних задач, для деяких систем будуть додаватися й інші показники, однак обрані показники якості залишаться основними. Ці показники є загальними, бо характеризують якість відтворення, виявлення подій контролю та якість функціонування об'єкта контролю. У даній роботі розглядається варіант співвідношення цих показників з уніфікованими показниками якості загальних стандартів якості програмного забезпечення. Обмеження для атрибутів характеристик формуємо на підставі аналізу нормативних документів для програмних систем даного типу.

Отже, введені в розгляд характеристики якості являються універсальними для програмного забезпечення даного класу інформаційних систем, бо

характеризують якість основних комплексів програм, із яких складаються ці системи.

Властиві програмному забезпеченню АСК характеристики точності відтворення параметрів і контролю допусків при виявленні подій, визначаються с залученням метрик, заданих у числовому виді. Тому, у даній роботі пропонується виділити функціональність як базовий показник якості критичних систем цільового призначення, до яких відноситься клас АСК. Висока питома вага даного показника забезпечить готовність програмного забезпечення до виконання очікуваних дій у зв'язку з призначенням в процесі експлуатації. Таким чином, показники функціональності та надійності є базовими показниками якості програмних систем, що оцінюють стан об'єктів контролю.

Після визначення всіх елементів моделі переходять до сертифікаційних випробувань, в ході яких обчислюються фактичні значення атрибутів підхарактеристик якості. Для критичних елементів атрибутів вони порівнюються з нормативними обмеженнями.

Якщо отримані фактичні значення показників якості відповідають нормативним вимогам, то подальшу оцінку можна провести, використовуючи інтегральний показник якості, в якому вага критичного показника повинна бути більше суми ваг другорядних. Дійсно, інтегральний показник якості можна застосувати для вибору кращого програмного забезпечення із декількох конкуруючих, за умови врахування важливості критичних характеристик, вимоги до яких повинні виконуватися обов'язково. Але, оскільки в моделі є показники з різними метриками, такими, як неперервні числові, бальні, якісні та інші, необхідно попередньо провести узгодження та нормування метрик. Це можна зробити, наприклад, шляхом уведення шкал для якісних та категорійних критеріїв і заданням вагових множників.

Оцінювати загальний рівень якості програмного забезпечення АСК пропонується в балах.

Поняття критичних показників якості допомагає остаточно вирішити проблему побудови узагальненої моделі якості, оскільки програмне забезпечення АСК для конкретної галузі повинне, принаймні, мати набір критичних властивостей, що включаються в побудовану модель якості для заданої області функціонування.

Крім того, запропонований спосіб оцінки критичних показників усуває проблему неоднорідності характеристик якості, оскільки кожен атрибут якості пропонується оцінювати в три етапи:

1) досягнення мінімальної задовільної планки рівня якості;

2) класифікація наявності властивості;

3) застосування вагових коефіцієнтів.

При цьому, на першому етапі можуть застосовуватися різні шкали та способи розрахунку метрик, а на другому етапі повинна бути використана тільки номінальна метрика. Впровадження третього етапу дозволяє зробити критичні показники незалежними від другорядних, тому що ваговий коефіцієнт кожного критичного показника пропонується задавати таким, щоб він перевершував суму балів усіх другорядних елементів атрибутів з моделі якості. В додаток до цього, третій етап дозволяє оцінювати ступінь перевищення мінімального задовільного значення для кожного критичного елемента атрибута шляхом пропорційного збільшення вагового коефіцієнта в залежності від значення, що було отримане на першому етапі.

Загальні вимоги до програмного забезпечення добре погоджуються з характеристикою функціональність, підхарактеристика повнота функцій. Наявність у повному обсязі властивостей мобільності, інтероперабельності, підтримки національного алфавіту та задоволення вимогам відкритих систем не впливає на безпеку життєдіяльності об'єкта контролю, який контролюється розглянутим критичним програмним забезпеченням. Тому цю групу характеристик віднесемо до другорядних показників. Вимоги до документації також є досить загальними і можуть бути перевірені простим порівнянням.

На основі аналізу вимог до програмного забезпечення контролю польотів здійснимо вибір показників якості.

Нормативна модель вимог ПЗ АСКП (ДСТУ 3275-95)

Вимога	**Атрибут вимоги**	**Інструкція**	**Обмеження**
Виконання переліку обов'язкових задач (x_1)	Відтворення параметричної інформації (x_{11})	Графічне представлення АП і РК (u_{15})	Синхронізація відображення (B_{x11})
...	...	...	...
Точність відтворення параметрів (x_7)	Інформація не менш ніж для 12 АП і 12 РК (x_{71})	Урахування дискретності опитування (u_{71})	Реєстраційний метод (так/ні) (B_{x71})
Вірогідність результатів контролю (x_8)	Оцінка помилок I роду (x_{81})	Критерій Неймана-Пірсона (u_8)	Розрахунковий метод E1<0.01 (B_{x81})
...	...	...	...

Відображення

Загальна модель якості ISO/IEC 9126

Характеристика	**Підхарактеристика**	**Атрибут**	**Елемент атрибуту**	**Метрика**
Функціональність (C_1)	Функціональна повнота (S_{11})	?	?	?
Функціональність (C_1)	Точність (S_{12})	...	...	...
...	...	...	...	...
Функціональність (C_1)	Захищеність (S_{14})	...	...	...
...	...	...	...	...
Надійність (C_2) або (y_9)	Безвідмовність (S_{21}) або (y_{91})	...	...	...
...	...	...	...	...

Вибірка, сполучення, узгодження

Сертифікаційна модель якості ПЗ АСКП

Характеристика	**Підхарактеристика**	**Атрибут**	**Елемент атрибуту**	**Метрика**	**Обмеження**
Функціональність (C_1)	Функціональна повнота (S_{11})	Відтворення параметричної інформації (A_{111})	Графічне представлення АП і РК (E_{1115})	Зовнішня номінальна метрика M_{1115}	Синхронізація відображення (B_{ce11})
...	...	...	...	...	...
Функціональність (C_1)	Точність (S_{12})	Точність відтворення параметрів (A_{121})	Інформація не менш, чим для 12 АП і 12 РК (E_{1211})	Номінальна метрика M_{1211}	Реєстраційний метод (так/ні) (B_{ce71})
...	...	...	...	...	...
Функціональність (C_1)	Точність (S_{12})	Вірогідність результатів контролю (A_{122})	Оцінка помилок I роду (E_{1221})	Метрична абсолютна шкала M_{1221}	Розрахунковий метод E1<0.01 (B_{ce81})
...	...	...	...	...	...
Надійність (C_2)	Безвідмовність (S_{21})	Частота відмов (A_{211})	Модель Шика-Волвертона (E_{2111})	Абсолютна шкала M_{2111}	Статистичні дані: X=|A1-A2|/B (B_{ce91})
...	...	...	...	...	...

Рис. 2.3. Відображення вимог до програмного забезпечення контролю згідно ISO/IEC 9126

Враховуючи вищесказане, пропонується класифікувати нормовані галузеві вимоги, здійснюючи їх відображення на характеристики моделі якості ISO/IEC 9126 (2001), як показано на рис.2.3.

Приступимо до узгодження показників і побудуємо сертифікаційну модель якості програмного забезпечення АСКП. Підсумовуючи критичні показники робимо висновок, що нижня планка узагальненого показника якості програмного забезпечення у розглянутому прикладі дорівнює 540 балам. Таким чином, для визнання програмного забезпечення контролю польотів таким, котре задовольняє вимогам замовника і нормативних документів, його показник якості повинний бути не менш ніж 540 балів. Використовуючи методику рангування, і, враховуючи 12 балів елементів другорядних атрибутів,одержимо наступну таблицю.

Таблиця 2.1

Оцінка рівня якості програмного забезпечення АСКП

Значення узагальненого показника (рівня) якості	**Оцінка програмного забезпечення**
$U_{ПЗАСКП}$ ′ 540	незадовільно
540 ≤ $U_{ПЗАСКП}$ ′ 546	задовільно
546 ≤ $U_{ПЗАСКП}$ ′ 552	добре
$U_{ПЗАСКП}$ 5 552	відмінно

Природно, що для позитивного рішення про сертифікацію й допуск до експлуатації, оцінка програмного забезпечення повинна бути не гіршою ніж “задовільно”.

Отримані результати можуть бути застосовані, також, розробниками для створення власної системи управління якістю програмного забезпечення автоматизованих систем контролю, як одного з класів інформаційних систем.

2.3 Опис технології оцінювання характеристик якості програмного забезпечення

Після того як побудована узгоджена модель якості ПЗ, проводяться сертифікаційні випробування, мета яких полягає у визначенні фактичних показників якості для характеристик, включених у модель. Для цього розробляється програма випробувань, створюються ТНД, проводиться тестування та обчислення показників якості. Процес сертифікації є трудомісткою й витратною процедурою, а тому, в цілях підвищення ефективності випробувань, скорочення витрат і отримання достовірних результатів, необхідно її автоматизувати. Для підтримки процедури сертифікації пропонується створити технологію сертифікаційних випробувань. Схема сертифікації буде мати вигляд, показаний на рис. 2.4.

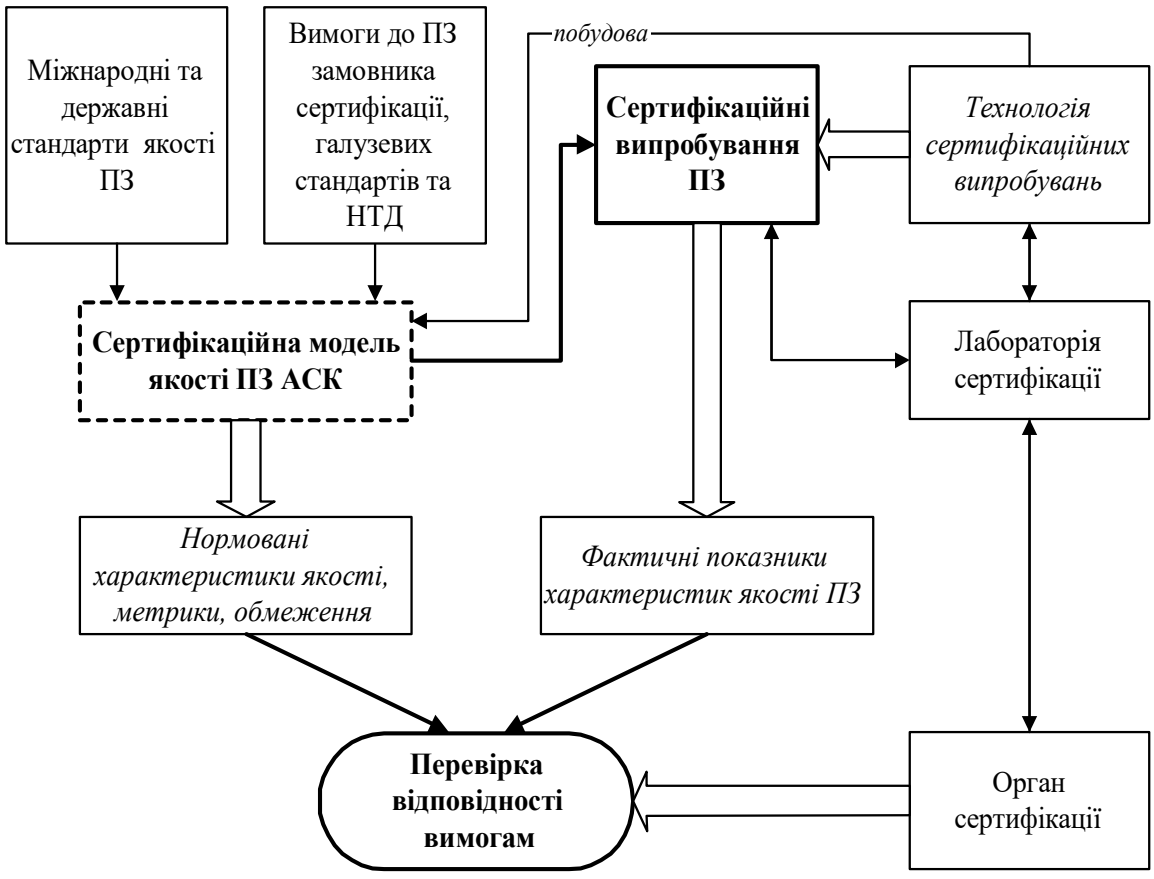

Рис. 2.4. Технологічна схема сертифікації програмного забезпечення

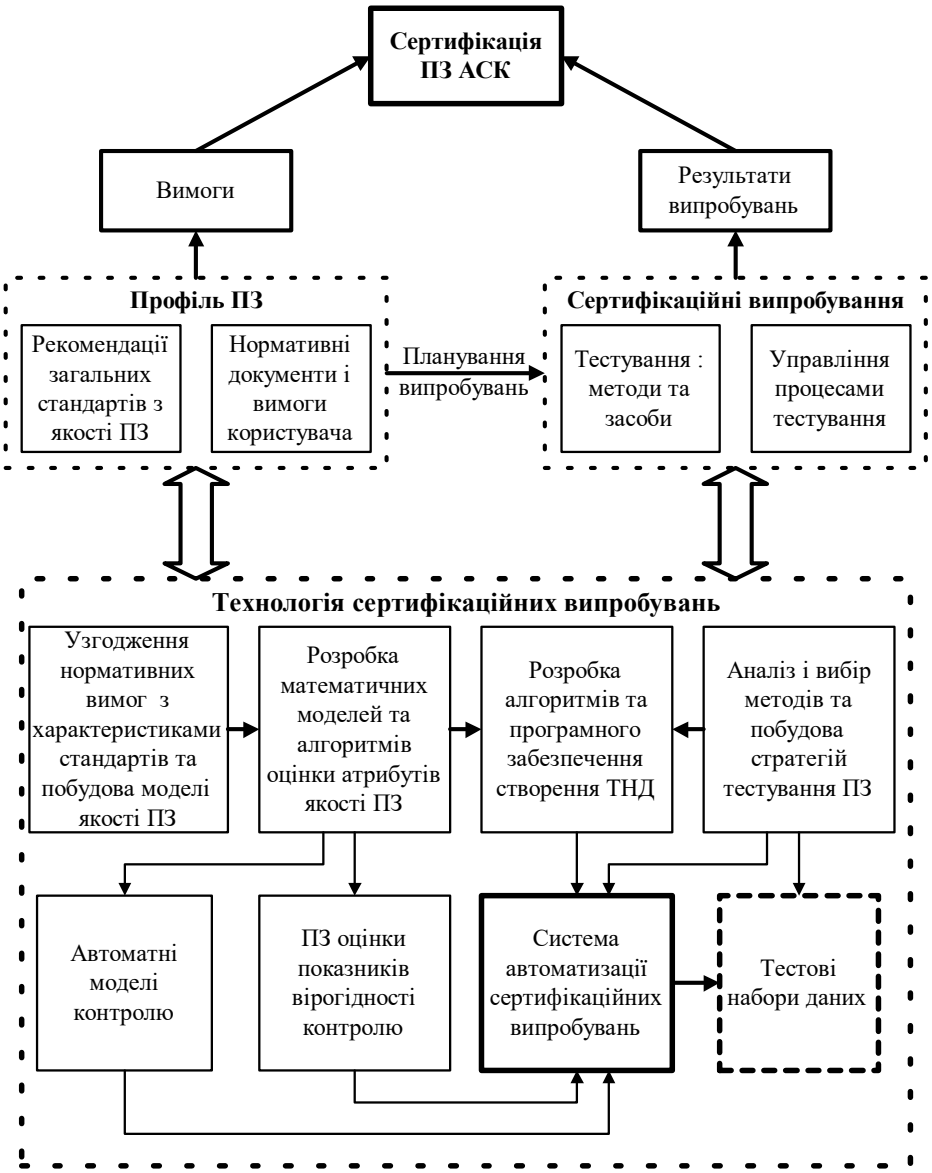

Рис. 2.5. Функціональна схема технології сертифікаційних випробувань програмного забезпечення АСК

Визначимо технологію оцінювання характеристик якості програмного забезпечення АСК при сертифікаційних випробуваннях як таку, що складається з наступних етапів:

1. Аналіз стандартів та формування вимог до програмного забезпечення АСК. Виділення належних характеристик якості на основі узгодження їх із вимогами до програмного забезпечення і побудова сертифікаційної моделі якості.

2. Вибір математичних методів і побудова моделей визначення фактичних показників якості ПЗ із подальшою розробкою алгоритмів їхнього розрахунку.
3. Розробка стратегій і методик тестування програмного забезпечення систем автоматизованого контролю, а також алгоритмів та програмного забезпечення генерації тестових наборів даних.
4. Створення системи автоматизації сертифікаційних випробувань програмного забезпечення систем автоматизованого контролю.

Функціональна організація випробувань програмного забезпечення показана на рис. 2.5.

Описану технологію можна застосувати для випробувань широкого класу автоматизованих систем оцінки стану об'єктів, алгоритми контролю яких мають вид логічних функцій з обмеженнями на контрольовані параметри й тривалістю в часі, тобто, для автоматизованих систем контролю динамічних або стаціонарних об'єктів, стан яких оцінюється за допомогою вимірювальної інформації (програмного забезпечення систем контролю та діагностичних систем).

2.4 Аналіз методів обчислення значень показників якості програмного забезпечення

Для перевірки відповідності ПЗ вимогам необхідно визначити фактичні значення показників якості, що можна зробити шляхом тестування. Методів тестування розроблено багато, вони об'єднуються в групи, кожна з яких призначена для певних цілей. Для вирішення другої проблеми сертифікації (обчислення фактичних показників якості програмного забезпечення АСК) придатні динамічні методи тестування, які полягають у виконанні програми на ЕОМ на спеціально згенерованих вхідних даних. Проблема полягає у виборі стратегій тестування та виду тестової інформації. Необхідно, також,

автоматизувати тестування, що забезпечить необхідну вірогідність результатів сертифікаційних випробувань і зменшить вартість їхнього проведення.

2.4.1 Огляд методів, які можна використати для тестування програмного забезпечення АСК

На практиці застосовують ряд різних категорій тестів, що відрізняються цілями, методами перевірки та оцінки результатів. Для вирішення проблеми вибору пропонується провести аналіз існуючих методів тестування програмних модулів і комплексів з урахуванням спеціалізації програмного забезпечення АСК. Розробники програмного забезпечення АСК проводять валідацію і верифікацію та досить серйозне тестування, що включає лексичний та синтаксичний аналіз, а також аналіз семантичної правильності як окремих модулів, так і комплексів у цілому. Тому в процесі сертифікаційних випробувань варто проводити поглиблене тестування програмних комплексів з урахуванням специфіки предметної області даного програмного забезпечення.

Вхідною областю даних для програмного забезпечення АСК є параметрична інформація. Наприклад, для програмного забезпечення АСКП в якості параметричної інформації виступає польотна інформація, яка представляє собою складно-організовані структуровані взаємозалежні дані. Ця інформація відображає динаміку поведінки ЛА як керованого об'єкта, рух якого має складну природу й описується системою диференціальних рівнянь. Звідси випливає, що для побудови ТНД с залученням методу імітаційного моделювання, необхідно моделювати динаміку руху, управляючі впливи й ситуації порушення обмежень, для чого створюється спеціалізований моделюючий стенд. Однак, виготовлення моделюючих стендів і підготовка ТНД за допомогою таких стендів є трудомісткою, потребуючою серйозних матеріальних вкладень, задачею.

Уникнути моделювання поведінки об'єкта контролю дозволяє розроблений у даній роботі підхід до створення ТНД, який заснований на модифікації штатної параметричної інформації. Цей підхід полягає в

моделюванні контрольованих подій, тобто в імітації різного роду відхилень параметрів управління й технічного стану об'єкта у штатній параметричній копії польоту.

Для тестування програмного забезпечення АСК можуть використовуватися різні методи. Найбільш формалізованим методом перевірки програмного забезпечення є символічне тестування. При використанні цього методу як еталон беруться правила структурної побудови програмних модулів, а перевірка виконання цих правил проводиться за допомогою аналізу тексту програм на алгоритмічній мові програмування. Метод символічного тестування може бути поглиблений аж до рівня формальної верифікації й доказу правильності програм.

Тому перевагу слід віддати динамічному тестуванню, що полягає у виконанні тестів програмами, приведеними до виду завантажувальних модулів. Серед динамічних методів особливо складним є метод детермінованого тестування, який дозволяє виявити відхилення в результатах роботи програми від еталонних значень, позначених у специфікаціях.

При тестуванні програмного забезпечення можна застосувати різні динамічні методи тестування. У даному розділі виконано другий і третій етапи технології, тобто визначено методи й стратегії, що найбільш ефективні при тестуванні модулів і комплексів програмного забезпечення АСК, та обґрунтовано вибір базового методу при створенні ТНД – методу імітаційного моделювання параметричної інформації.

2.4.2 Загальна характеристика комплексів програм АСК та особливості їх тестування

Програмні комплекси мають складну структуру, а тому доцільно проводити тестування поетапно (наприклад, модуль → програма → комплекс). Переваги розбивки процесу тестування на етапи виражаються в наступному:

1) полегшується виявлення невідповідностей через невеликий обсяг модуля;
2) з'являється можливість використання комбінаторики тестування.

Якщо при перевірці складних систем не виявлено недоліків, то з великою долею ймовірності можна вважати, що тестові випробування проводилися тривіально. Результати недбалого тестування програмного забезпечення АСК можуть бути досить серйозні. Оскільки програмні комплекси, що оцінюють стан об'єкта контролю, є критичними, то помилкові результати їх роботи можуть привести до катастрофічних наслідків.

Визначимо методи перевірки основних програмних комплексів, що входять до складу програмного забезпечення АСК, та побудуємо стратегії і методику їх випробувань для перевірки відповідності програмного забезпечення зовнішнім вимогам. Розглянемо характеристики програмного забезпечення АСКП, що представлені на рис. 2.6.

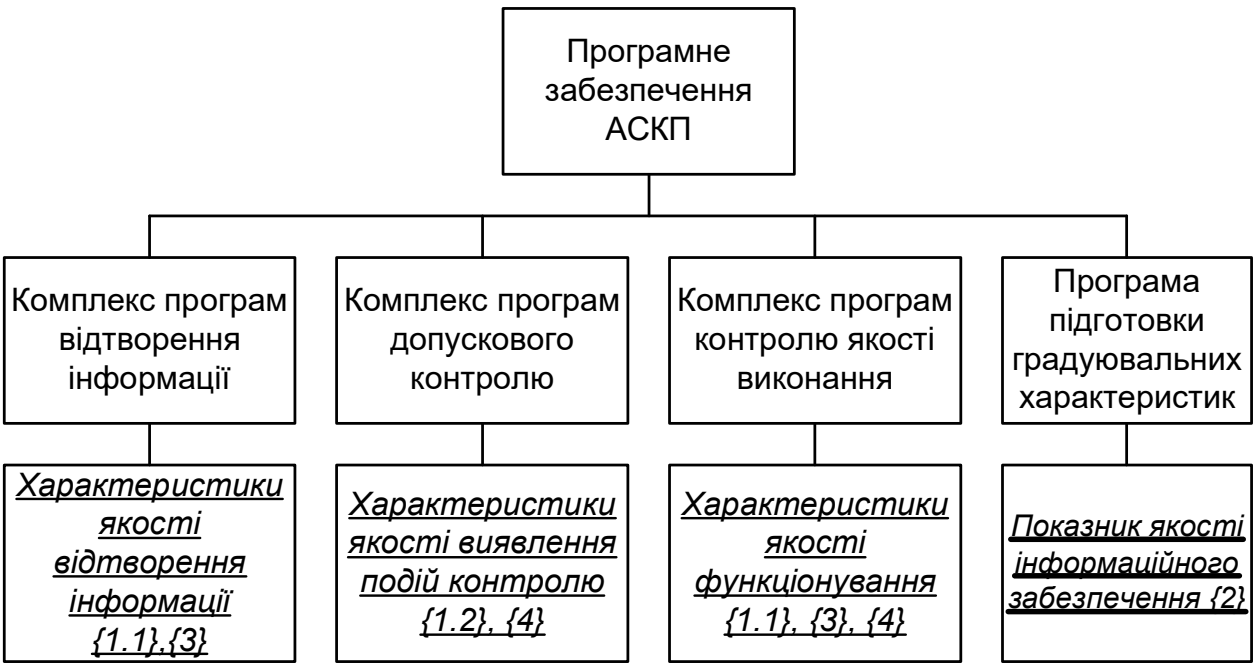

Рис. 2.6. Характеристики якості комплексів контролю

Хоча вхідною інформацією для кожного з комплексів контролю служить інформація параметричних реєстраторів, однак у кожнім з них реалізовані різні класи алгоритмів, для оцінки яких застосовуються різні показники якості, що, у свою чергу, визначає різні вимоги до методів тестування і використовуваним ТНД.

У комплексі програм відтворення реалізовані алгоритми попередньої обробки інформації і перетворення кодових значень параметрів у фізичні (показник якості {2}). Але основними вимогами до виконуваних функцій тут є забезпечення заданої точності відтворення (показники якості {1.1}, {3}). Тестовими даними для перевірки такого комплексу можуть бути довільні послідовності значень із припустимої області. У цьому випадку для задання ТНД зручно використовувати значення функцій, які можна легко обчислити, або заздалегідь підготовлені еталонні вибірки вхідних величин із відомою формою графічного представлення.

У комплексі допускового контролю реалізовані алгоритми обчислення складних логічних функцій, визначених на множині неперервних параметрів, що описують траєкторію руху динамічної системи. Тому ТНД повинні містити інформацію, що відповідає реальному процесу, а також моделювати процес із порушеннями. З метою моделювання виходів за обмеження необхідно робити корекцію інформації штатного процесу. Задання конкретних значень параметрів у моделі при внесенні виходів за обмеження повинне здійснюватися з урахуванням забезпечення необхідної вірогідності результатів контролю {4}, показники якої для допускового контролю є основними показниками якості. Також, в цілях повноти виявлення небезпечних відхилень, важливим являється показник якості {1.2}.

Комплекс програм контролю якості виконання польоту реалізує алгоритми відтворення значень параметрів у контрольованих точках польоту і на його ділянках (подібно комплексу відтворення) із метою обчислення показників якості виконання елементів польоту, а також алгоритми пошуку контрольованих ситуацій у копії польоту (подібно комплексу допускового контролю). У даному випадку в якості ТНД можна використовувати штатну тестову копію з повним описом інформації, а також відкоректовану (модельну) копію з внесеними ситуаціями виходів за обмеження, які відповідають різним значенням показників пілотування. Контрольованими показниками якості тут

є забезпечення заданої точності відтворення {1.1}, {3} і вірогідності результатів контролю {4}.

2.4.3 Аналіз існуючих динамічних методів тестування та дослідження можливості їх застосування

Тестування дозволяє знайти помилки в програмах і програмних комплексах, а також невідповідності між даними, що згенеровані згідно вхідних специфікацій, і результуючими даними, які не потрапили до припустимих вихідних специфікацій. У випадку відсутності невідповідностей і помилок на множині припустимих даних, програмне забезпечення визнається працездатним. Тестування є одним із самих трудомістких етапів життєвого циклу програмного забезпечення.

При проведенні сертифікаційних випробувань програмного забезпечення АСК, тестування дозволяє розмежувати придатні й непридатні програмні продукти, що є особливо важливим фактором, який впливає на безпеку функціонування об'єкта контролю. Зауважимо, що ефективність тестування безпосередньо впливає на якість ПС.

Виникає необхідність дослідження відомих методів з метою вибору з існуючого спектра тих методів, які будуть найбільш корисні для тестування програмного забезпечення АСК. По кожному з методів, що найкраще служать цілям створення ТНД, необхідно привести обґрунтування і дати висновок.

Дослідимо можливість застосування **стратегії "чорного ящика"** для тестування програмного забезпечення АСК. При використанні цієї стратегії програма розглядається як "чорний ящик", на вхід якого подаються тестові набори, а на виході проводиться оцінка: чи не відбулося відхилень по вихідним даним програми від описаних у її специфікаціях.

Для покриття різних маршрутів виконання в програмах логічної обробки, надзвичайно ефективним є метод **еквівалентних розбивок**. Розробка ТНД, яка заснована на методі еквівалентних розбивок, проводиться у два етапи:

1) виділення класів еквівалентності;

2) побудова тестів.

Правильні класи еквівалентності містять правильні вхідні дані програми (акцентують увагу на правильних умовах), а неправильні – неправильні (акцентують увагу на неправильних умовах).

Метод еквівалентних розбивок враховує синтаксичні й семантичні правила побудови програм, а також спосіб задання даних, і тому він ефективний для тестування програмного забезпечення АСК. В якості ілюстрації розглянемо виділення класів еквівалентності для програмного забезпечення АСКП, де слід керуватися наступними правилами :

1) Якщо вхідна умова визначає область значень (наприклад - висота геометрична виміряється в діапазоні від 0 до 750 м), то виділяється один правильний клас еквівалентності ($0 \leq$ висота геометрична ≤ 750) і два неправильних (висота геометрична < 0 і висота геометрична > 750).
2) У випадку, якщо вхідна умова визначає число значень (наприклад, кількість аналогових параметрів, видаваних на графік у процесі автоматизованої обробки може бути від 1 до 12), то задається один правильний клас еквівалентності і два неправильних (жодного, більш 12).
3) Якщо вхідна умова описує множину вхідних значень і відомо, що кожне значення програма обробляє особливим чином (наприклад, відомі наступні 2 типи разових команд: “звичайні” – 8 РК у двох байтах, “УКР-4” – 4 РК у двох байтах), то визначається правильний клас еквівалентності для кожного значення з множини й один неправильний клас для значення поза множиною (наприклад, тип разових команд “розширені” – 12 РК).

Можливість застосування методу еквівалентних розбивок для тестування програм логічної обробки ілюструє наступний приклад.

Нехай програма, що входить до складу комплексу програм допускового контролю, визначає подію контролю: “Перевищення максимальної експлуатаційної швидкості польоту для літака”. Подія має номер 73 згідно каталогу алгоритмів контролю і записується у виді наступного логічної умови: **S073**=$(7120 \leq H \leq 10300) \wedge V/588$.

У даному випадку можна визначити два правильних класи (1, 4) і три неправильних класи еквівалентності (2, 3, 5), як показано в табл. 2.2.

Таблиця 2.2.

Визначення класів еквівалентності при тестуванні програмного забезпечення АСКП

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Обмеження на висоту польоту	Висота барометрична більше чи дорівнює 7120м і менше чи дорівнює 10300м $7120 \le H \le 10300$ (1)	$H < 7120$ (2) $H > 10300$ (3)
Обмеження на швидкість польоту	Швидкість приладова більше чи дорівнює 588 км/год $V \ge 588$ (4)	$V < 588$ (5)

Якщо в цьому прикладі підготувати дані для класів 2, 3 і 5, то можна з'ясувати поводження програми при обробці неправильних класів еквівалентності, а для перевірки поводження програми при виконанні предикатів скористаємося тестами по правильних класах еквівалентності 1 і 4.

Для задач відтворення не рекомендується виділяти класи еквівалентності, оскільки тут необхідно забезпечити подачу вхідної інформації на всьому припустимому діапазоні зміни параметрів. Однак у модулях комплексів програм допускового контролю і контролю стану та якості функціонування об'єкта основними розв'язуваними задачами є процеси логічної обробки, в яких маршрути виконання істотно залежать від вхідних даних. З метою покриття маршрутів пропонується виділяти різні класи еквівалентності по вхідним і вихідним даним, причому для кожного режиму роботи об'єкта будується свій клас еквівалентності.

Практика тестування дозволяє зробити висновок: застосовуючи тести, що враховують граничні умови, ми одержуємо більше шансів виявити помилки в модулях, незалежно від їхньої функціональної приналежності.

Метод аналізу граничних значень є логічним розвитком методу еквівалентних розбивок. Підготовка ТНД (на прикладі тестування модулів АСКП) зводиться до включення граничних умов, для чого в тести вводяться дані, що розташовані безпосередньо вище чи нижче границь вхідних чи вихідних класів еквівалентності.

При побудові тестів по методу аналізу граничних значень для комплексів контролю будемо керуватися наступними правилами:

1) Для умов, що описують область значень, потрібно будувати тести для границь області, а також для випадків незначного виходу за границі області (включаючи правильні й неправильні класи еквівалентності). Наприклад, правильна область вхідних значень укладена в інтервалі [0,1]. У цьому випадку пишуться тести для значень: 0, 1, 0.001, -0.001, 1.001, 0.999.
2) У випадку, якщо умова описує дискретний ряд значень, генеруються тести для мінімального й максимального значень, а також для значень більших і менших мінімального й максимального. Наприклад, мінімальна кількість АП, видаваних на графік у комплексах програм відтворення, дорівнює 1, а максимальна дорівнює 24. Тоді генеруються тести для наступних значень: 1, 24, 0, 2, 23, 25.
3) Для кожної з вихідних умов варто застосувати правило 1. Наприклад, якщо значення вертикальної складової швидкості дорівнює 1м/с, то бажано побудувати тести для значень 1.01 м/с і 0.99 м/с.
4) Застосувати правило 2 для кожної вихідної умови. Так, якщо на виході комплексу програм відтворення, на графік видається від 1 до 12 разових команд, то тести готуються для значень: 0,1,2,11,12,13.

Метод аналізу граничних значень рекомендується широко використовувати для тестування програмного забезпечення АСК, як продовження методу еквівалентних розбивок.

Метод функціональних діаграм характеризується, в основному, набором алгоритмічних правил для вибору значень операндів в операторах програми і визначенням очікуваних вихідних значень для кожного застосовуваного тесту

(стовпці таблиці рішень). Метод корисний для визначення побічних наслідків, що приводять до помилок, але є громіздким у реалізації, і тому не рекомендується для тестування програмного забезпечення АСК.

При використанні *метода припущення про помилку* спочатку складається список можливих помилок, а також особливих ситуацій, у яких приблизно можуть з'явитися помилки, а потім складаються відповідні ТНД. У визначенні можливих помилок може послужити поглиблений аналіз специфікацій (визначаються моменти, що не описані в специфікаціях).

Припустимо, що необхідно скласти набори даних для тестування програми фільтрації, що входить у підсистему попередньої обробки параметричної інформації. Визначимо ситуації згідно методу припущення про помилку :

1) Набір вхідних даних для параметра, що підлягає фільтрації, порожній.
2) Набори вхідних даних для параметра містять той самий ряд значень.
3) Набір вхідних даних із копії параметричної інформації вже був відфільтрований.
4) Набір вхідних даних містить викиди за областю припустимих значень.

Для більш ефективного застосування методу необхідно детально описувати область передбачуваних значень вихідних даних. Метод простий у реалізації і рекомендується для створення ТНД при випробуваннях програмного забезпечення АСК.

Дослідимо, тепер, можливість застосування **стратегії “білого ящика”**. Для проведення тестування по стратегії “білого ящика”, необхідно мати тексти програм на мові високого рівня, а тестові дані одержуються шляхом аналізу логічної організації програм. Вичерпному вхідному тестуванню по стратегії “чорного ящика” може бути поставлене у відповідність вичерпне тестування маршрутів по стратегії “білого ящика”, відповідно до якого вважається, що програма цілком протестована, якщо при проходженні тестів вдається здійснити виконання програми по всіх можливих маршрутах її графа передач управління. При оцінці методу вичерпного тестування маршрутів виявляється,

що число не повторяючих один одного маршрутів навіть у простій програмі обчислюється астрономічною величиною (в основному, через застосування циклів у програмах). Але навіть після проведення вичерпного тестування маршрутів програма може містити помилки.

Продуктивним представляється підхід, що сполучає корисні можливості стратегій “білого” і “чорного ящика”. При тестуванні програмного забезпечення АСК корисним буде застосування різновидів методу вичерпного тестування маршрутів для тестування, налагодження і проведення випробувань комплексів програм допускового контролю в частині логіки визначення подій контролю, оскільки даний метод дає гарні результати при створенні тестів по перевірці логіки програм.

Тестування програм по стратегії “білого ящика” ґрунтується на покритті операторів, покритті рішень, покритті умов, чи на комбінаторному покритті умов програми. Усі ці методи, узяті в цілому, покривають логіку програми. Чим вище ступінь покриття логіки програми, тим більш всебічними вважаються ТНД.

Критерій покриття всіляких маршрутів можна замінити критерієм виконання кожного оператора програми хоча б один раз, однак і цей критерій є недостатнім для задовільного тестування. Ще більш сильним є критерій покриття умов, що накладає додаткову вимогу: результати кожної умови в рішенні повинні виконуватися при тестуванні не менш одного разу. Критерій покриття умов не завжди задовольняє критерію покриття рішень, тому часто застосовують критерій покриття рішень/умов, що поєднує два попередніх методи. Останній критерій має недолік: за допомогою цього критерію не можна перевірити виконання всіх результатів умов, оскільки в практичному програмуванні деякі умови бувають сховані іншими умовами. Цю проблему вирішує метод комбінаторного покриття умов, у якому потрібно створення такої кількості тестів, щоб усілякі комбінації результатів умови в кожнім рішенні, включаючи всі точки входу, виконувалися хоча б один раз. Для

програм, які містять рішення, що мають більш однієї умови, мінімальний набір тестів повинний спричинити виконання всіх комбінацій результатів умов у кожнім рішенні.

2.4.4 Формування і оцінка стратегій тестування модулів програмного забезпечення АСК

Оскільки вичерпне тестування у відповідності зі стратегіями “білого” і “чорного ящиків” є складним, для побудови тестів використовуються різні методи, кожний з який належить до першої, або до другої стратегії. Для побудови більш вдалих тестів рекомендується використовувати ці методи в комплексі, оскільки кожний з них має свої переваги й недоліки (наприклад, властивість виявляти одні типи помилок і пропускати інші).

Опишемо *стратегію тестування модулів програмного забезпечення АСК*. Оскільки модулі з комплексів відтворення в основному займаються візуалізацією параметричної інформації, при їхньому тестуванні в меншому ступені застосовні методи аналізу граничних значень і стохастичного тестування, що добре “працюють” для модулів з комплексів допускового контролю і контролю якості функціонування об'єкта поблизу границь інтервалу допуску, тому що ці комплекси повинні якісно визначати події контролю. Через те, що модулі з комплексів допускового контролю реалізують алгоритми контролю, при їхньому тестуванні мало що може дати метод припущення про помилку (тут він не створює нових умов у порівнянні з еквівалентною розбивкою). Цей метод може бути вдало застосований при тестуванні модулів з комплексу відтворення.

Зауважимо, що при тестуванні програм допускового контролю маршрути виконання істотно залежать від послідовності вхідних даних. Так, для послідовності даних, що складаються з набору параметрів, що належать одним класам еквівалентності, відповідає один маршрут виконання, а набір елементів з інших класів визначає інший маршрут виконання програми.

Аналіз методів тестування, що відносяться до стратегій “білого” і “чорного” ящиків, із метою визначення кращих методів при тестуванні програм, які входять до складу основних комплексів контролю, дав результати приведені в табл. 2.3.

Таблиця 2.3.

Характеристика методів тестування модулів контролю

Методи тестування	Модулі комплексу відтворення	Модулі комплексу допускового контролю	Модулі комплексу контролю якості функціонування
Еквівалентна розбивка	–	+	+
Аналіз граничних значень	–	+	+
Припущення про помилку	+	–	+
Стохастичне тестування	–	+	+
Покриття рішень/умов	–	+	+
Комбінаторне покриття умов	+	+	+

Ніякий з вищерозглянутих методів не може забезпечити повного тестування модуля, а може лише забезпечити створення набору тестів, що покривають кожний свою область вхідних і вихідних значень та логіки цього модуля. Тому пропонується об'єднати розглянуті методи створення тестів і визначити стратегію тестування модулів контролю, що буде складатися з наступних загальних правил:

1) Використовуючи метод еквівалентної розбивки, виділити правильні й неправильні класи еквівалентності по вхідним і вихідним даним. Метод еквівалентної розбивки є ефективним для тестування будь-яких модулів і програм, що входять до складу програмного забезпечення АСК.
2) Обов'язково використати метод граничних значень для аналізу граничних значень по вхідним і вихідним змінним програми. Аналіз граничних значень може дати додаткові набори тестових умов.

3) Логіка програми перевіряється згідно критеріїв покриття рішень, покриття умов чи комбінаторного покриття умов, із яких останній є найбільш повним.
4) Для тестування модулів і програм допускового контролю і контролю якості функціонування об'єкта, слід застосувати метод динамічного стохастичного тестування, який рекомендується використати усередині підобластей і особливо поблизу границь.
5) Додаткові ТНД одержимо, використовуючи метод припущення про помилку.

Розглянемо більш детально особливості *застосування класів еквівалентності* при тестуванні програмного забезпечення АСК. Алгоритми контролю (події) складаються з умов. Кожен предикат займає одну позицію в алгоритмі контролю, а кожна умова є логічною функцією від, як правило, більш ніж одного предиката. Для довільної події контролю $\mathbf{S_k}$ можна визначити класи еквівалентності, причому для кожного предиката визначається свій клас еквівалентності.

Оскільки розглянуті комплекси записуються на мовах програмування високого рівня, то умови контролю кодуються за допомогою операторів if, if ... then, if ... then ... else, do ... while, і while. Правильність таких операторів на 99 відсотків залежить від правильності написання умов. Таким чином, якщо умови записані вірно, то для правильних вхідних даних (наборів елементів із правильних вхідних класів еквівалентності) одержимо правильні результати (очікувані дані з правильних вихідних класів еквівалентності). Звідси випливає, що метод тестування покриття умов (1) із стратегії "білого ящика" сполучається з методом еквівалентна розбивка (2) (стратегія "чорного ящика"). Ці методи взаємно доповнюють один одного.

Розглянемо, тепер, можливість використання *моделей детектабільності* для оцінки стратегій тестування. Оскільки для тестування ПС застосовують різні стратегії, то необхідно дослідити питання формування критеріїв вибору найкращих методів тестування.

До надійності програмного забезпечення критичних систем висуваються вимоги на порядок вищі, ніж вимоги до програмного забезпечення загального призначення. Однак, надійність програмного забезпечення високоцілісних систем через їхню складність далеко не завжди має задовільні показники і немає повної впевненості, що система не містить помилок, оскільки традиційне практичне тестування не в змозі забезпечити необхідний рівень якості.

Акцентуємо увагу на наступних проблемах:

1) відсутність достовірної інформації про те наскільки ефективно ті чи інші застосовані методи тестування виявляють помилки;
2) наскільки одна стратегія тестування краще (ефективніше) іншої.

На практиці для оцінки ефективності застосованих методів, як правило, використовують евристичні підходи. Для вирішення проблеми вибору найкращих методів і стратегій з існуючого спектра використаємо поняття детектабільності, тобто здатності цих методів до виявлення помилок, або невідповідностей специфікаціям. Цей інструмент пропонується використати для визначення того факту наскільки один метод краще іншого. Як міру використовуємо Р-міру, тобто ймовірність виявлення хоча б одного дефекту в програмі.

2.4.5 Побудова стратегій тестування комплексів програм АСК

Перехід до великих програм і програмних систем, які містять від тисяч до десятків тисяч операторів, вимагає застосування спеціального набору способів і розбивки процесу проведення тестових випробувань на етапи. Дотепер був розглянутий перший етап тестування програмних систем - тестування модулів. Наразі необхідно розглянути другий етап - тестування комплексів програм.

В порівнянні з тестуванням модулів тестування комплексів АСК має свої особливості, і відрізняється від простого тестування модулів через складність і різноманіття зв'язків між модулями і підсистемами, що входять до складу

комплексу. При тестуванні комплексів програм варто спільно використовувати стратегію і методики тестування, викладені вище.

Складемо *стратегію попередньої оцінки* програмних комплексів АСК. Наступні загальні категорії тестів необхідно створювати при тестуванні комплексів програм контролю відповідно до вимог стандартів :

1) тести комплексу програм на відповідність його специфікаціям;
2) тести оцінки повноти переліку функцій, виконуваних комплексом;
3) тести перевірки логіки модулів і тести по методу аналізу граничних значень;
4) тести для перевірки міжмодульних інтерфейсів;
5) оцінка функціонування програм у критичних умовах;

Згрупуємо ці категорії у п'ять етапів стратегії попередньої оцінки комплексів АСК.

Очевидно, що на першому етапі варто перевірити даний комплекс на відповідність його специфікаціям, які повинні містити точний опис поводження програм, що складають комплекс. Мета тестування специфікацій полягає в перевірці функціональної повноти модулів різних ієрархічних рівнів, а також у визначенні відповідності результатів, отриманих при тестових випробуваннях, вихідним даним, описаним у специфікаціях. Якщо інформація на входах і виходах програмних модулів, взаємодіючих між собою в складі комплексу програм, однозначно відповідає один одному відповідно до своїх специфікацій, то будемо вважати, що програми задовольняють вимогам специфікацій.

Як другий етап при тестуванні комплексів програм виступає оцінка повноти переліку функцій, виконуваних комплексом. Для визначення повноти функцій необхідно скласти тестові набори даних по кожній з розв'язуваних комплексом задач, забезпечити проходження кожного тесту й оцінити результати тестування. У випадку задовільної оцінки комплекс програм вважається функціонально повним.

Наступним етапом є тестування модулів і програм. Основною метою цього тестування є перевірка якості обробки інформації, що надходить на вхід модулів, для чого здійснюється оцінка інформації на їхньому виході. Крім того, при тестуванні модулів, логіка кожної програми перевіряється по методу комбінаторного покриття умов, а оцінка вхідним і вихідним класам еквівалентності дається згідно методу аналізу граничних значень.

Четвертим етапом є тестування функціональних груп програм, що виконується з метою перевірки інформаційних і управляючих зв'язків між модулями і програмами, що входять до складу комплексу. Тут виконується тестування міжмодульних та людино-машинних інтерфейсів. Для тестування структури підсистем і шляхів проходження інформації рекомендується скористатися детермінованим динамічним тестуванням із застосуванням стохастичного підходу.

П'ятий етап полягає в тестуванні комплексу в цілому, для чого виконуються :

- оцінка комплексу на відповідність технічній документації;
- перевірка поводження комплексу на граничних (критичних) наборах даних;
- оцінка витрат ресурсів (обсяг оперативної пам'яті, завантаження ЦП і т.п.).

На додаток до вищенаведеної стратегії необхідно проводити поглиблене тестування комплексів програм контролю з урахуванням їх специфіки. Засобами поглибленого тестування можуть служити методи стохастичного динамічного тестування, або тестування комплексів програмного забезпечення АСК із застосуванням імітаційно-моделюючого стенда.

Опишемо *метод імітаційного моделювання параметричної інформації* для тестування програмного забезпечення АСК. При тестуванні комплексів програм контролю можна успішно застосовувати принципи динамічного тестування в реальному часі, оскільки характеристики динамічного тестування підходять для тестування комплексів АСК.

Зазначимо, що при тестуванні комплексів потрібно задавати велику кількість вихідних даних складної структури та еталонних значень для оцінки

результатів, а тому для генерації ТНД використовують програмні моделі й імітатори.

Для створення модельних ТНД при тестуванні комплексів АСК можна застосувати також імітаційно-моделюючі стенди. Однак цей процес вимагає великих матеріальних вкладень, тому що для імітації реальної параметричної інформації необхідна розробка спеціальної апаратури, оскільки ця інформація являє собою складно-організовані структуровані взаємозалежні дані. Для динамічних об'єктів ця інформація відображає динаміку поводження керованого об'єкта, рух якого має складну природу й описується системою диференціальних рівнянь. Звідси випливає, що для побудови ТНД необхідно моделювати динаміку руху, керуючі впливи, ситуації порушення обмежень і записувати інформацію в прийнятому форматі с залученням спеціалізованого моделюючого стенда.

В якості прикладу розглянемо АСКП, для яких фактор реального часу не є визначальним, однак, істотним є вид тестової інформації, яка повинна імітувати реальну ПІ. При тестуванні модулів відтворення АП і РК, можна використовувати заздалегідь підготовлені еталонні вибірки вхідних величин із відомою формою графічного представлення. Однак, при тестуванні модулів допускового контролю, де оцінюються групи взаємозалежних фізичних параметрів у динаміці їхньої зміни, найбільш рентабельним рішенням для створення ТНД є використання реальної параметричної інформації, зареєстрованої на борті ЛА.

Виготовлення апаратури, що імітує ПІ, наприклад, у виді набору датчиків-імітаторів, керованих деяким контролером, являється непростою задачею. Тому пропонується інший підхід: замість генерації інформації за допомогою апаратури, для імітації параметричної інформації використати штатні записи параметричних реєстраторів, виконані під час контролю функціонування об'єкта.

Необхідно створити систему генерації тестового набору даних та модуль обробки результатів тестування, якы будуть входити в якості основних

компонентів у систему сертифікаційних випробувань програмного забезпечення АСК (рис. 2.7).

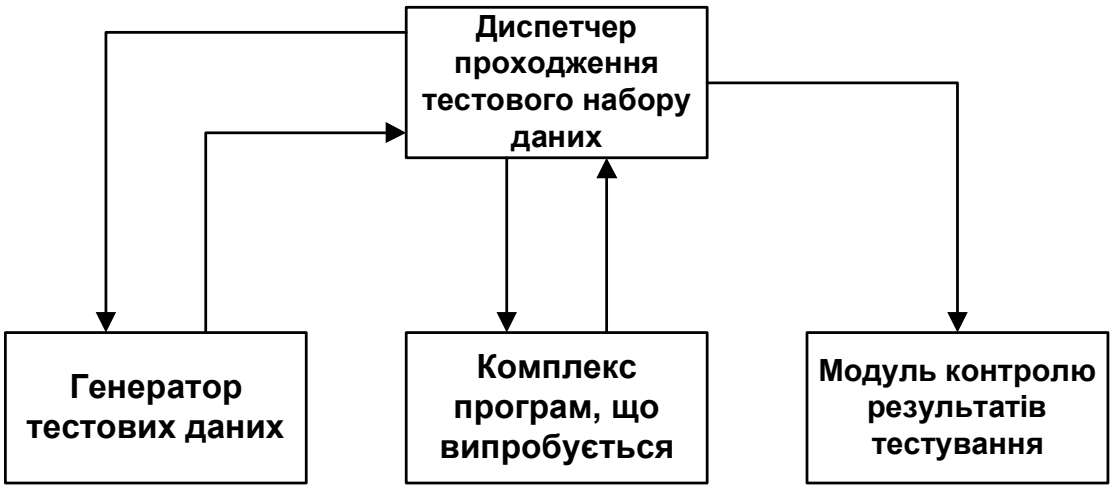

Рис. 2.7. Функціональна організація процесу тестування та генерації ТНД і обробка результатів тестування

Тестування комплексів програм контролю пропонується проводити відповідно до методики динамічного комплексного тестування з пріоритетним застосуванням методу імітаційного моделювання параметричної інформації, що складається з наступних правил :

1) Для створення ТНД використовується технологічна ЕОМ із спеціальним програмним забезпеченням – системою автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів. Оскільки обробка параметричної інформації (попереднє оброблення, відтворення, допусковий контроль та ін.) може проводитися поза реальним часом, тестування комплексів програм АСК проводиться також поза реальним часом.
2) Вхідною областю даних для програмного забезпечення АСК, які тестуються, є інформація, записана в реальному часі на борті ЛА під час польоту.
3) Виходи за обмеження моделюються в штатних копіях польотів за допомогою програмного імітатора – комплексу проектування та генерації

ТНД, що входить до складу системи автоматизації сертифікаційних випробувань програмного забезпечення АСК, і використовується, також, для оцінки результатів тестування.

4) Комплекс проектування та генерації ТНД забезпечує редагування копій польотів різних видів ЛА, записаних різними типами бортових реєстраторів.
5) Оцінка результатів тестових випробувань виконується за допомогою системи автоматизації сертифікаційних випробувань на технологічному комп'ютері за результатами якості відтворення, вірогідності виявлених подій контролю, якості попередньої обробки та іншим показникам.

Висновки до розділу

В даному розділі описано концепцію побудови сертифікаційної моделі якості ПЗ АСК, суть якої полягає у відображенні вимог до ПЗ на характеристики загальної моделі якості, що сформована на основі рекомендацій стандартів ISO/IEC. Концепція дозволяє отримати уніфіковані методи оцінки атрибутів якості. Досліджено правила побудови узагальненої моделі якості ПЗ АСК із використанням інтегральної оцінки, що дає можливість порівняння конкуруючих програмних продуктів у даній галузі.

РОЗДІЛ 3.
РЕАЛІЗАЦІЯ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА СЕРТИФІКАЦІЇ ПРОГРАМНИХ РІШЕНЬ ДЛЯ АВТОМАТИЗОВАНИХ СИСТЕМ

Метою імітаційного моделювання випробувань є визначення фактичних показників якості програмного забезпечення АСК. В основному, випробування проводяться шляхом тестування, яке є трудомісткою процедурою. Скорочення термінів сертифікаційних випробувань, зменшення витрат і підвищення вірогідності результатів можна досягти шляхом автоматизації. Автоматизуються найбільш трудомісткі операції, такі як створення тестових наборів даних і обробка результатів тестування, що дозволяє підвищити ефективність тестування і, як наслідок, рівень якості програмного забезпечення.

Застосування засобів автоматизації (третя проблема сертифікації) забезпечує система автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів, яка складається з програмно-апаратного імітаційно-моделюючого стенда і спеціалізованого програмного забезпечення, яке використовується для підготовки ТНД. Систему автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів рекомендується використовувати, також, розробникам комплексів програм контролю польотів для забезпечення етапів тестування й налагодження.

3.1 Реалізація функціональної схеми автоматизації сертифікаційних випробувань програмного забезпечення

Система автоматизації сертифікаційних випробувань складається з трьох функціональних автоматизованих комплексів, які взаємозалежні по управлінню та інформації, що оброблюється, і потребують участі експерта. Організація комплексів системи та зв'язки між ними показані на рис. 3.1.

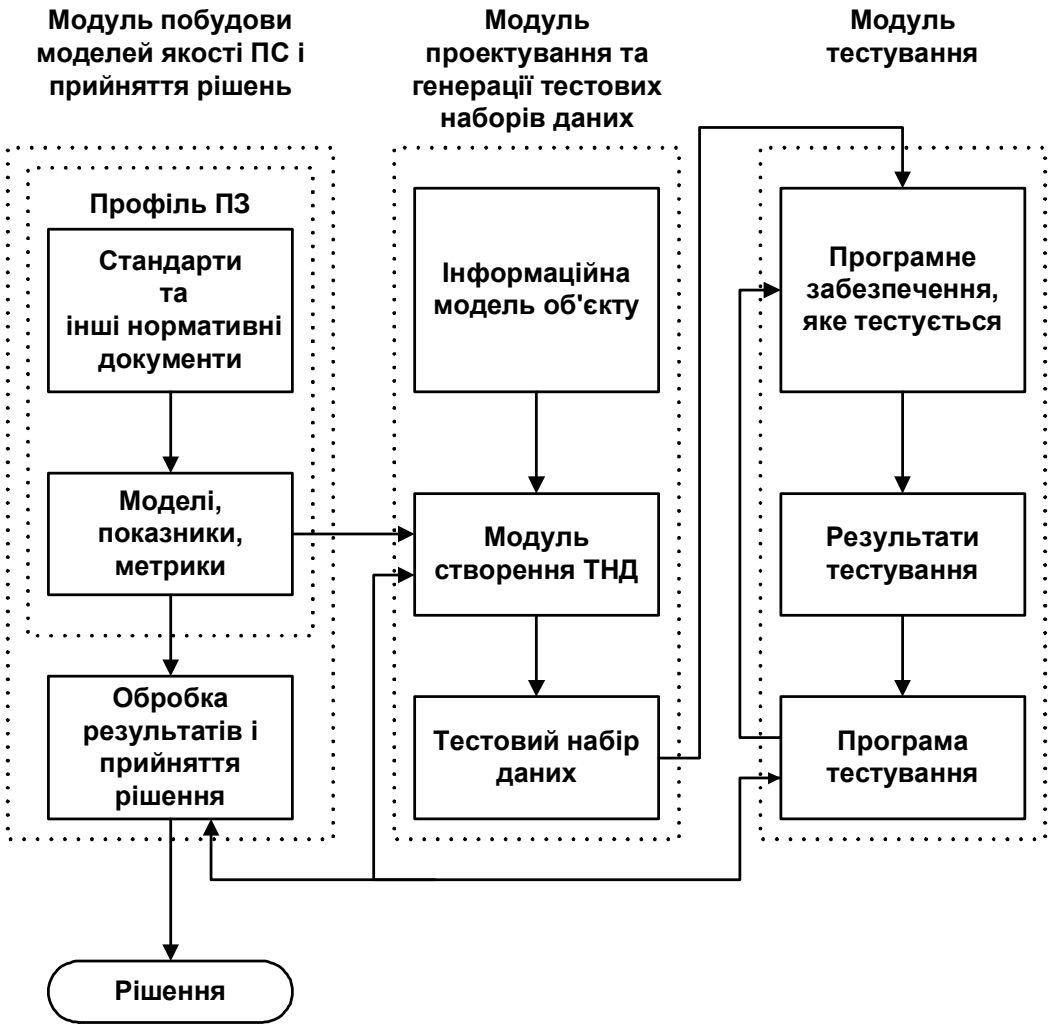

Рис. 3.1. Функціональна організація процесу отримання рішення в системі автоматизації випробувань програмного забезпечення

Розглянемо призначення і функції кожного з модулів.

У модулі побудови моделей якості ПС і прийняття рішень на основі аналізу вимог діючих стандартів і нормативних документів формується модель якості системи, яка включає нормовані властивості, характеристики й атрибути, котрі визначають ці властивості, та метрики для їх обчислення.

При проведенні сертифікації програмного забезпечення визначальними являються рекомендації державних стандартів, а також вимоги галузевих стандартів та нормативних документів. Вимоги останніх відображають вузький, галузевий погляд на якість і містять, в основному, вимоги до нормованих характеристик програмного забезпечення. Для того щоб

результати сертифікаційних випробувань у повній мірі відображали якість, і могли використовуватися не лише для допуску програмного забезпечення до експлуатації, а також для порівняння його з конкуруючими продуктами, у дійсній роботі пропонується в процесі побудови сертифікаційної моделі класифікувати характеристики галузевих стандартів згідно характеристик міжнародних стандартів, узгодивши конфліктуючі характеристики.

Таким чином, модель якості буде включати в себе:

1) основні характеристики якості ПС, які перевіряються при проведенні випробувань (для програмного забезпечення АСКП - це показники {1.1},{1.2},{2},{3},{4} і т.д.);
2) перелік критеріїв, за допомогою яких перевіряється відповідність програмного забезпечення АСК показникам, зазначеним у п.1).

Основними показниками якості ПС є:

1) функціональність, що задається для НСАО ПІ у виді переліку розв'язуваних задач і точності відтворення параметричної інформації;
2) надійність, яка для НСАО ПІ втілюється в підхарактеристику безвідмовність, тому що неправильне рішення про стан об’єкта, викликане низькою достовірністю допускового контролю, еквівалентне відмові.

Підхарактеристики якості виділяються зі стандартів, а критерії беруться з нормативно-технічної документації. Для компонентів нормованих вимог, що сформульовані кількісно (вірогідність результатів контролю, точність результатів вимірів параметрів і ін.), задаються алгоритми обчислення необхідних показників. В цьому ж комплексі містяться алгоритми обчислення числових характеристик, метрики визначення не числових характеристик, алгоритми обробки результатів випробувань, та відповідне програмне забезпечення.

Функціональний блок “Обробка результатів і прийняття рішення” є результуючим. У ньому на основі обробки результатів тестування, одержуваних із комплексу тестування, група експертів приймає рішення про

відповідність ПС вимогам. Аналіз результатів виконується на автоматизованому робочому місці експерта за допомогою спеціального сервісного програмного забезпечення.

Для визначення зовнішніх характеристик якості проводиться тестування програмного забезпечення. Для цього необхідно організувати генерацію тестових наборів даних, склад яких визначається програмою тестування.

Модуль проектування та генерації тестових наборів даних служить для генерації оптимальних тестових наборів даних, що забезпечують тестування програмного забезпечення, а також для запису створених ТНД на носії у форматі параметричних реєстраторів. Вхідною інформацією для комплексу є інформаційні копії польотів ЛА даного типу. Далі в копію польоту вносяться такі корективи параметрів, щоб вона містила порушення пілотування й відхилення в роботі систем ЛА, що повинні контролюватися НСАО ПІ.

Створені в такий спосіб ТНД надходять у комплекс тестування. У цьому комплексі по заздалегідь розроблених методах і стратегіях виконуються тестові випробування програмного забезпечення АСКП. Результати випробувань надходять у комплекс побудови моделей якості ПС і прийняття рішень, де на основі автоматизованої обробки результатів тестування експерти отримують фактичні значення показників якості, що входять до сертифікаційної моделі.

3.2 Реалізація модуля генерації тестових наборів даних

Програмне забезпечення, що призначене для створення ТНД відповідно до нормативного документа, складається з комплексу проектування та генерації тестових наборів даних і ряду допоміжних підсистем і програм. Графічний комплекс генерації ТНД надає інструментарій для побудови ТНД на базі штатних копій ПІ, даючи можливість шляхом імітаційного моделювання модифікувати копію польоту таким чином, щоб у ній були представлені польотні ситуації, які потрібно контролювати.

3.2.1 Склад і характеристика модуля генерації ТНД

Програмний комплекс генерації ТНД розроблений у відповідності з об'єктно-орієнтованим підходом до проектування програмних продуктів, який дозволяє адекватно моделювати поведінку реальних об'єктів (проблемна область, сценарії взаємодії з експертом, динамічне керування обробкою даних і т.д.). За допомогою класів описані не тільки меню і віконний інтерфейс, але також структура кадрів бортових реєстраторів, алгоритми контролю польотів, циклограми ЛА, їх градуювальні характеристики й ін. Комплекс реалізований у стандарті мови C++.

Програмний комплекс генерації ТНД забезпечує поліпшений інтерфейс користувача, що включає дружній меню-орієнтований інтерактивний режим роботи, засоби віконного діалогу і багатовіконного перегляду в стилі Windows. Він надає широкий спектр сервісних можливостей, у який, крім функцій керування процесом випробувань, включені наступні функції:

1. визначення етапів польоту і пошуку моментів контролю;
2. створення й модифікування файлів градуювальних характеристик;
3. конвертації файлів копій польотів з ущільненої форми в пряму і назад;
4. контролю розпізнавальних даних;
5. візуалізації і корекції файлів із копіями польотів ЛА;
6. створення й запису тестових наборів даних;
7. накопичення інформації про результати випробувань.

3.2.2 Структура модуля генерації ТНД

Графічний комплекс генерації ТНД входить до складу системи автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів і забезпечує візуалізацію, перегляд, аналіз і внесення змін у копію польоту, зареєстровану бортовими реєстраторами. Програми, що входять до складу комплексу, написані мовою C++.

Комплекс простий в експлуатації, максимально повно забезпечений системою меню й підказок, містить у собі декілька ступенів контролю й аналізу вірогідності даних, що вводяться, з інформуванням користувача про причини помилок. Функціонально комплекс організований у виді десяти взаємозалежних підсистем:

1. управляюча;
2. фільтрації;
3. візуалізації;
4. робочих профілів;
5. підготовки файлів градуювальних характеристик;
6. підготовки файлів циклограм;
7. тестування логіки алгоритмів контролю;
8. обчислення вірогідності подій контролю.

Структурна схема зв'язків між підсистемами представлена на рис. 3.2.

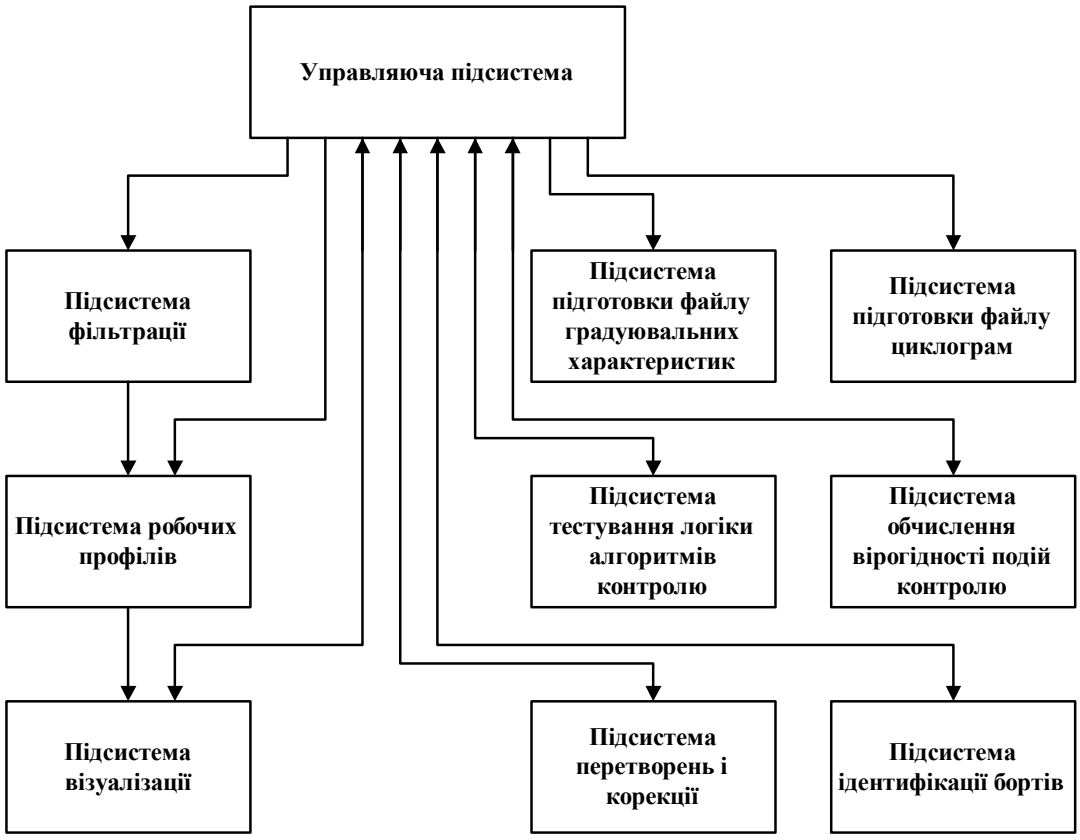

Рис. 3.2. Структура модуля генерації ТНД

Управляюча підсистема – це програми, що забезпечують користувальницький інтерфейс у системі сертифікації програмного забезпечення контролю польотів. За допомогою розвиненої організації віконного інтерфейсу і системи управляючих меню, підсистема забезпечує настроювання й виконання всіх інших підсистем, на вибір користувача, а також автоматичне визначення форматів управляючих файлів і файлів даних. Композиційна гнучкість програм, що складають управляючу підсистему, дозволяє легко підключати будь-яку кількість нових підсистем перетворень і корекції, а також, у випадку потреби, виключати які-небудь компоненти. Програмна реалізація управляючої підсистеми заснована на класі TmainWindow, який є головним і викликає функції підпорядкованих класів.

Клас TEditBinary – призначений для представлення даних про обрані користувачем РК, а також для їхнього редагування у випадку вибору такої опції. Даний клас є класом підсистеми візуалізації.

Клас TchooseButton – використовується для представлення даних про обрані користувачем АП, а також, як і у випадку з РК, дозволяє звертатися до підсистеми редагування для зміни обраних параметрів. Даний клас також є класом підсистеми візуалізації .

Клас TTask – призначений для збереження управляючих даних. Ці дані дозволяють коректно й у зручному для користувача виді здійснювати вивод на екран дисплея. Перелік даних, що зберігаються в даному класі перерахований нижче:

- назва параметра й вид (аналоговий параметр чи разова команда);
- канал реєстрації і тип, що дозволяє визначити яким способом буде обчислюватися фізичне значення АП, чи як буде витягатися РК.
- початкове значення «голки» (піксела) промальовування, її ціна й колір, яким буде виводитися даний параметр.

Клас TAddFDR є допоміжним класом. Він дозволяє модернізувати систему у випадку появи нового типу накопичувача. У діалоговому вікні виклику досить увести наступні дані:

- назва накопичувача і повний шлях до файлу динамічної бібліотеки, який дає можливість працювати з новим накопичувачем;
- розширення файлу копії для даного накопичувача.

Клас TEdBCDialog – призначений для зміни управляючих даних, зв'язаних із РК (назва параметра, колір, яким буде виводитися параметр, параметр візуалізації, що сигналізує про те чи буде даний параметр виводитися на екран).

Клас TTimeLine – це клас, що реалізує представлення даних про час у поточній копії польоту. Даний клас відноситься до підсистеми візуалізації, а його робота показана в головному вікні управляючої підсистеми (рис. 3.3).

Клас TMessageBar – призначений для виводу на екран службової інформації. Даний клас є стандартним класом бібліотеки Borland C++ і застосовується без внесення змін у його структуру.

Клас TEdAP – це породжений клас від класу TEdit, що є стандартним класом бібліотеки Borland. До основних поліпшень класу можна віднести можливість зміни числових значень у визначеному діапазоні, із їхнім автоматичним коректуванням у випадку виходу за межі діапазону, встановленого в кожному конкретному випадку.

Клас TAlgControl є універсальним класом для опису набору алгоритмів контролю польотів ЛА даного типу. Цей клас містить вид алгоритму контролю, предикати і логічні умови, обмеження, а також має посилання на елементи класу TReliability (допусковий і довірчий інтервал). Клас TAlgControl належить підсистемі тестування логіки алгоритмів контролю. У цій підсистемі за допомогою запрограмованих автоматних моделей контролю, здійснюється внесення в копію польоту потрібних подій контролю й створення тестових наборів даних, які покривають граф передач управління алгоритму по запиті експерта.

Клас TReliability містить інформацію про показники вірогідності подій контролю для ЛА даного типу, зокрема, математичне сподівання і дисперсію параметрів, допускові й довірчі інтервали. Клас належить підсистемі

обчислення вірогідності подій контролю. У цій підсистемі визначаються показники вірогідності шуканих подій контролю і мінімально можливі допуски на параметри (нормальний розподіл випадкової величини і погрішності засобів контролю при перевірці статистичних гіпотез згідно з критерієм Неймана-Пірсона). Підсистема реалізована на мові Object Pascal.

Вид головного вікна управляючої підсистеми представлений на рис. 3.3.

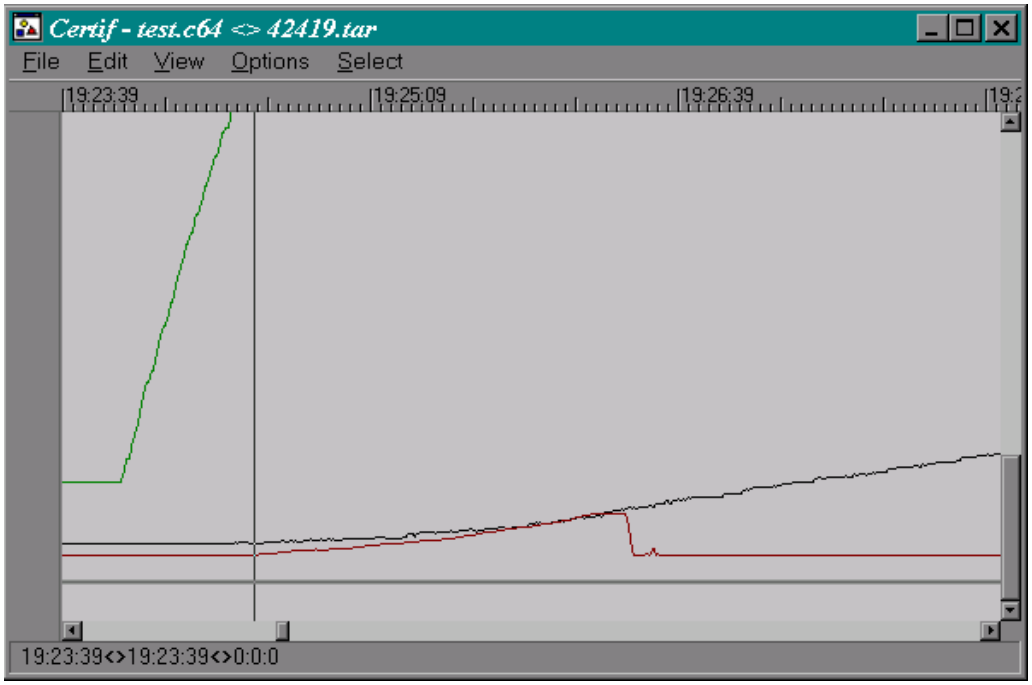

Рис. 3.3. Головне вікно системи генерації тестових наборів даних

Клас TViewGrafic є класом підсистеми візуалізації і призначений для представлення всієї копії польоту в зменшеному виді в одному вікні. Зовнішній вигляд роботи класу показаний на рис. 3.4.

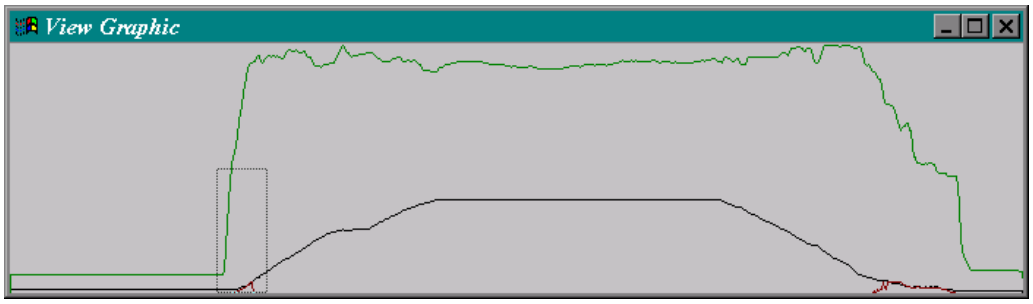

Рис. 3.4. Вікно представлення всієї копії у зменшеному вигляді

Підсистема робочих профілів реалізує користувальницький інтерфейс, який допомагає користувачу вибирати службові дані як для АП, так і для РК. Реалізація даної підсистеми здійснена на основі класів TEdAPDialog і TEdBCDialog. Клас TEdAPDialog реалізує діалогове вікно, яке дозволяє користувачу вибирати потрібні для перегляду й редагування АП.

Клас TEdAPDialog призначений для введення управляючих даних, зв'язаних з АП. Перелік службових даних, що вводиться, представлений нижче:

- назва параметра;
- початкове значення «голки» промальовування на графіку й ціна «голки»;
- колір, яким буде виводиться даний параметр та індикатор, що сигналізує про те чи буде параметр виводитися на екран.

Зовнішній вигляд роботи даного класу представлений на рис. 3.5.

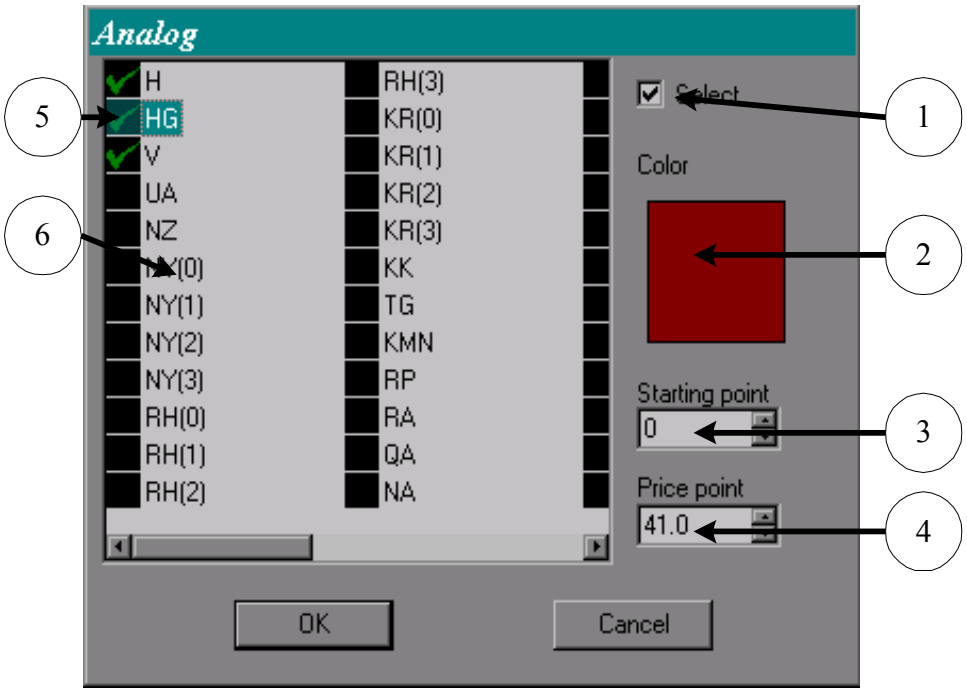

Рис. 3.5. Діалогове вікно вибору АП

1 – індикатор статусу стану поточного параметра;

2 – індикатор кольору поточного параметра;

3 – поле для редагування значення «початкової голки» параметра;

4 – поле редагування значення «ціни голки» поточного параметра;

5 – індикатор статусу стану параметра;

6 – індикатор назви параметра.

Клас TEdBCDialog, як і попередній, реалізує діалогове вікно, але для РК. Оскільки РК не потрібно масштабувати, то кількість параметрів, що вводяться, зменшено.

Підсистема перетворень і корекції надає користувачу широкі можливості по аналізу й коректуванню контрольованих параметрів польоту. Підсистема являє собою комплекс програм, що реалізує наступні функції:

1. перегляд копії в обох напрямках, у межах інтервалу часу, визначеного користувачем;
2. зручне настроювання параметрів графіка і виду АП і РК;
3. перегляд і коректування значень коду і фізики АП, а також значень РК у вікні коректування;
4. методи коректування інтервалу шляхом застосування алгоритмів виду “лінія”, “ламана”,“поліном” і “синусоїда”;
5. автоматична заміна змісту копії після підтвердження коректування графіка.

Підсистема перетворень і корекції копії польоту забезпечує зручне меню і засоби коректування контрольованих параметрів, надаючи інструментальні засоби для безпосередньої зміни даних обраної користувачем ділянки копії, а також, у випадку потреби, видалення деякої ділянки. Реалізація підсистеми виконується на основі класів TEdChAPDialog, TEdChPoint, TEditBinary.

Клас TEdChAPDialog – являє собою спеціалізований графічний редактор, що дозволяє редагувати обраний користувачем АП, із наданням засобів для заміни обраної ділянки на бажану. Перелік можливостей: лінія, синусоїда, трапеція (ламана), поліном. Зовнішній вигляд роботи даного класу представлений на рис. 3.6.

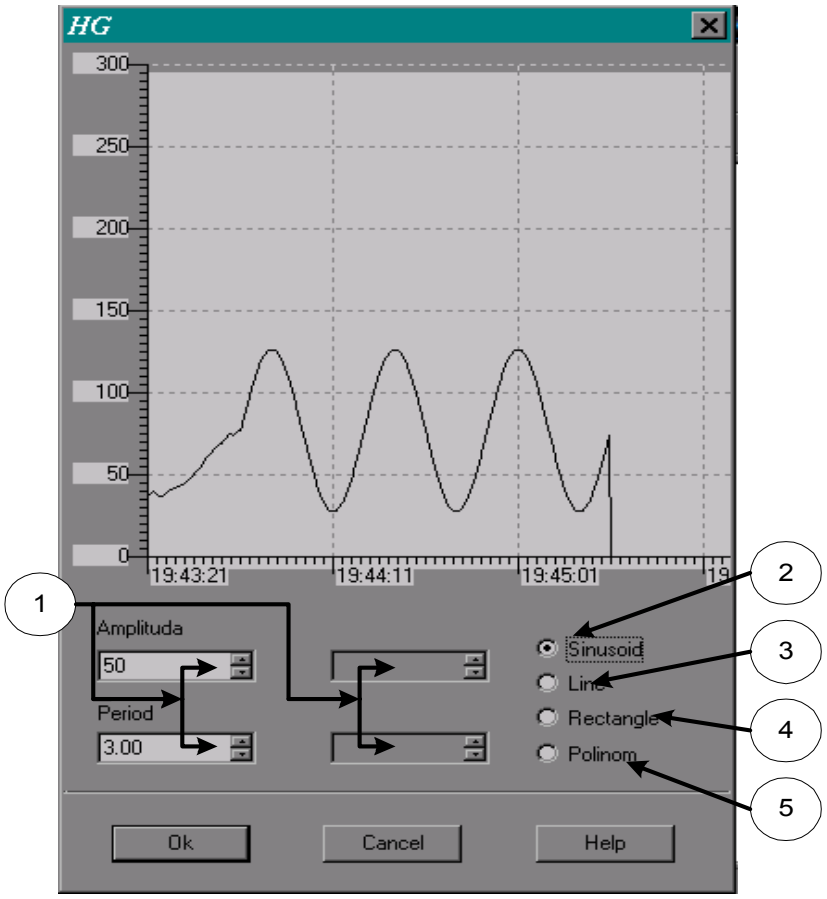

Рис. 3.6. Діалогове вікно редагування параметра

1 – поле редагування робочих даних;

2 – вибір режиму редагування у виді синусоїди;

3 – вибір режиму редагування у виді прямої;

4 – вибір режиму редагування у виді трапеції (ламана);

5 – вибір режиму у виді полінома (крива; наприклад, кубічна парабола).

Клас TEdChPoint являє собою невеликий спеціалізований редактор, призначений для редагування значення АП в одиничному інтервалі часу (у даній точці копії польоту). Введення даних виконується в числовому виді. Зовнішній вигляд роботи даного класу представлений на рис. 3.7.

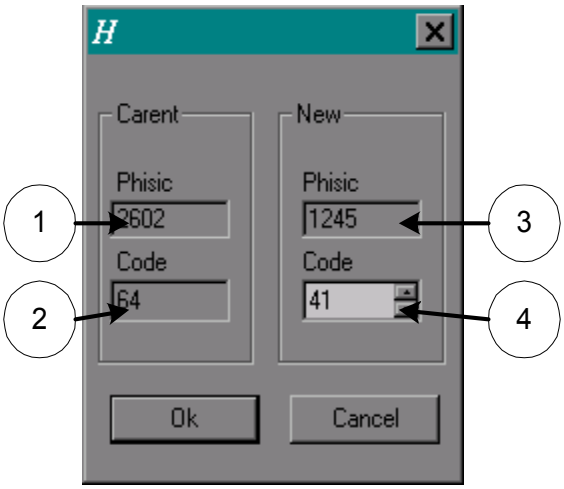

Рис. 3.7. Діалогове вікно редагування файлу копії

1 – поле індикації дійсного фізичного значення АП у поточній позиції;

2 – поле індикації дійсного кодового значення АП у поточній позиції;

3 –поле індикації модифікованого фізичного значення АП;

4 – поле індикації модифікованого кодового значення АП.

За допомогою системи автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів були створені сотні тестових копій параметричної інформації з внесеними в них подіями контролю.

3.3 Опис модулів системи автоматизації випробувань програмного забезпечення

3.3.1 Файли даних і управляючі файли

До набору файлів даних входять:

- файли, що містять копії польотів;
- файли градуювальних характеристик для кожного об’єкту.

Файл копії польоту являє собою двійковий файл, перетворений зі стислої (ущільненої) форми в повну. Файл містить розпізнавальні дані, коди аналогових параметрів, коди бінарних сигналів, розташовані по відповідних

каналах. Файл копії польоту може бути змінений підсистемою редагування копії для одержання ТНД.

Файл тарувальних характеристик містить у собі номера каналів АП і набір коефіцієнтів для інтерполяції кожного з каналів. Коефіцієнти використовуються в ряді програм підсистеми візуалізації й підсистеми редагування копії для перетворення кодового значення АП у фізичну величину. Файл градуювальних (тарувальних) характеристик генерується підсистемою підготовки файлу градуювальних характеристик на базі стандартного текстового файлу.

До набору управляючих файлів входять файли циклограм. Файл циклограми – це текстовий змінюваний файл, що містить імена й типи АП і РК, а також канали їхньої реєстрації. Номера каналів АП задаються в один рядок незалежно від того, чи є вони одноопитуваними, чи багатоопитуваними параметрами. Додатково до АП записується їхній тип. Це дає можливість визначити чи буде кодове значення параметра витягатися безпосередньо з копії, або розраховуватися по визначеному алгоритмові. У випадку разових команд задаються два значення: номер каналу і номер біта в цьому каналі. Крім того, як і у випадку з АП, записується тип параметра, що дозволяє визначити додаткову інформацію про РК.

Файл циклограми залежить тільки від типу польотного реєстратора й типу ЛА і може бути заготовлений заздалегідь, до запуску графічної системи, а також може бути відредагований підсистемою підготовки файлу циклограми.

3.3.2 Підсистема фільтрації

Підсистема фільтрації складається з ряду програм, що перетворюють вихідний файл копії з формату бортового реєстратора до формату системи сертифікації, і, після виконання різноманітних змін, можуть зберегти його у вихідному форматі файлу копії, а також здійснюють корекцію й маркірування збійних кадрів копії. Дане перетворення файлу копії приводить до прискорення обробки даних за рахунок максимального використання

системних можливостей Windows. Реалізація підсистеми здійснена на основі функцій InitBuf, Convertor і CompilerFile.

Функція InitBuf призначена для визначення розміру масиву, необхідного для вводу файлу, а також підрахунку кадрів і визначення збійних кадрів. Функція Convertor призначена для перетворення файлу копії у внутрішній формат системи, а також для маркірування збійних кадрів і фільтрації даних, зв'язаних із часом. Функція CompilerFile призначена для перетворення даних із внутрішнього формату системи назад у вихідний формат файлу прямої неущільненої копії.

3.3.3 Підсистема візуалізації

Підсистема візуалізації являє собою комплекс програм, що реалізують представлення даних файлу копії у зручному для перегляду й редагування виді. Підсистема візуалізації забезпечує видачу графічної, текстової і цифрової інформації на екран відеомонітора. Інформація видається у виді графіків аналогових параметрів і разових команд. Кожен параметр маркірується, масштабується і видається своїм кольором у вікно виводу графічної інформації.

Користувачу надані зручні діалогові можливості по підключенню АП і РК, вибору кольору, заданню початкової голки і ціни поділу. Досить кілька щигликів “мишею” щоб додати для візуалізації новий параметр. Графіки параметрів прорисовуються у вікні стислого представлення польоту (клас TviewGrafic, рис. 3.4) і у вікні реального виду польоту рис. 3.3).

У вікні стислого представлення приводиться інформація з копії польоту з проріджуванням таким чином, щоб весь політ від злету до посадки помістився в це вікно. За допомогою спеціального “повзунка” можна переміщатися по графіках у цьому вікні, причому той фрагмент, який можна спостерігати через “повзунок”, відображається у вікні реального виду польоту. У вікні реального виду прорисовані АП і РК таким чином, як вони записані у файлі копії польоту, причому розмір вікна точно відповідає “повзунку” у вікні “профілю”

польоту. У вікні реального виду промальована лінійка масштабу, а також відбита часова вісь, яка цифрується згідно з розпізнавальними даними часу з копії польоту.

Підсистема візуалізації реалізована на основі розроблених класів TeditBinary, TchooseButton, TviewGrafic, TtimeLine.

Клас TEditBinary – це клас, що реалізує представлення інформації про РК в окремому спеціалізованому вікні. Робота даного класу показана на рис. 3.8.

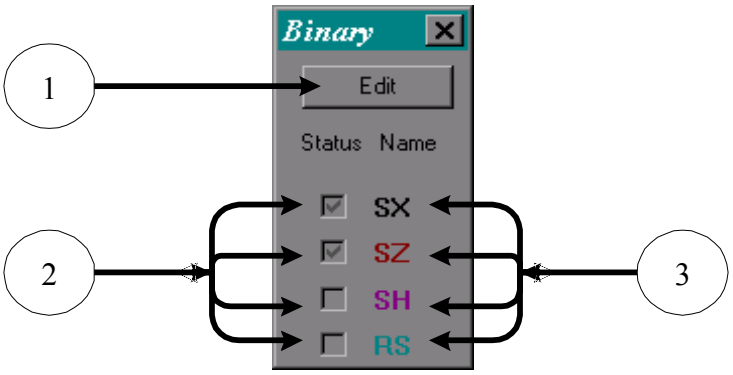

Рис. 3.8. Діалогове вікно класу TEditBinary

Тут:

1– кнопка, що призначена для відкриття діалогу редагування РК;

2– індикатори станів РК;

3– індикатори назв РК.

Клас TChooseButton, на відміну від попереднього класу, реалізує представлення інформації про АП, а також надає можливість виклику діалогового вікна, яке дозволяє здійснити редагування обраного параметра. Вид даного класу представлений на рис. 3.9.

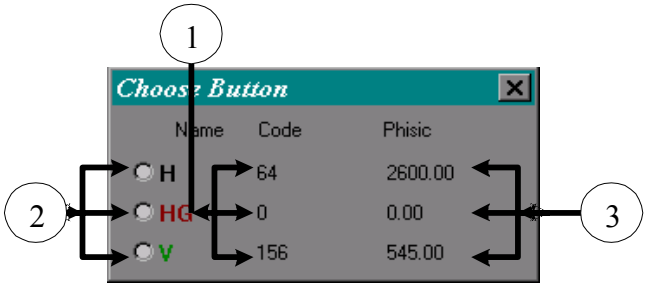

Рис. 3.9. Діалогове вікно класу TChooseButton

1- індикатори, що показують кодове значення відповідних параметрів;

2- назва АП, виділена відповідним кольором, і кнопки для опції редактора;

3- індикатори, що показують фізичні значення відповідних параметрів.

Клас TTimeLine реалізує представлення даних копії польоту, що містять інформацію про час польоту. Робота даного класу представлена на рис. 3.3.

3.3.4 Підсистема підготовки файлів градуювальних характеристик

Підсистема підготовки файлів градуювальних характеристик являє собою програмний засіб для створення (чи редагування наявних) файлів із тарувальними характеристиками. Реалізація підсистеми виконана на основі створеного класу TTarir. За допомогою цього класу реалізується зручний користувальницький інтерфейс, що функціонує як спеціалізований текстовий редактор. Зовнішній вигляд даної підсистеми представлений на рис. 3.10.

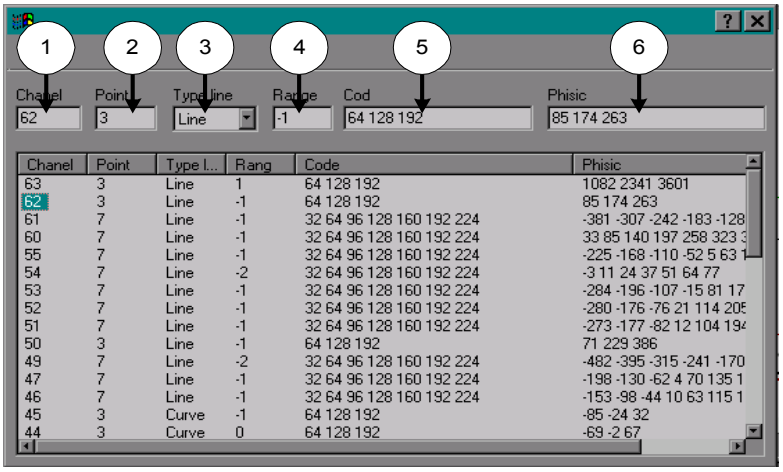

Рис. 3.10. Діалогове вікно редагування файлу градуювальних характеристик

Тут: 1 – поле редагування номера каналу; 2 – поле редагування кількості точок інтерполяції; 3 – поле вибору типу інтерполяції (апроксимації); 4 – поле редагування коефіцієнта з основою 10; 5 – поле редагування кодового значення; 6 – поле редагування фізичного значення.

3.3.5 Підсистема підготовки файлів циклограм

Підсистема підготовки файлів циклограм являє собою програмне забезпечення для створення (чи редагування наявних) файлів із циклограмами на різні типи ЛА і реєстраторів. Реалізація підсистеми здійснена на базі класу Tcyclogram, який забезпечує зручний інтерфейс, що функціонує як спеціалізований текстовий редактор, але з відмінною рисою: він не дозволяє редагувати які-небудь інші файли, що не мають відношення до файлів циклограм. Зовнішній вигляд даної підсистеми представлений на рис. 3.11.

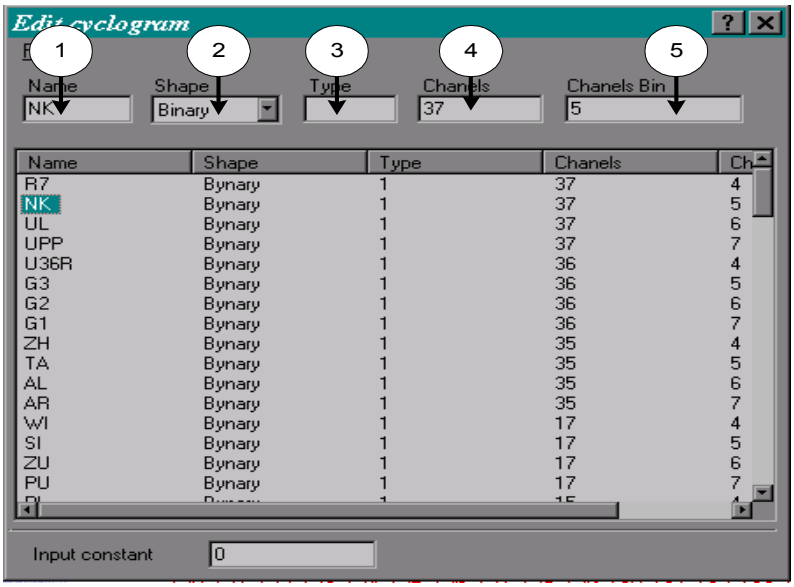

Рис. 3.11. Діалогове вікно редагування файлу циклограми

1 – поле редагування назви параметра;

2 – поле вибору виду параметра;

3 – поле редагування типу параметра;

4 – поле редагування номера каналу поточного параметра;

5 – поле редагування гілок поточного параметра.

3.3.6 Підсистема ідентифікації бортів

Підсистема ідентифікації бортів призначена для спрощення роботи управляючої підсистеми. Реалізація підсистеми здійснена на основі класу TIdenBort. Клас TIdenBort виконаний у вигляді діалогового вікна

спеціалізованого редактора. Користувачу надається можливість додавати, або змінювати вже існуючі значення. Перелік значень, що вводяться, наступний: тип ЛА; мінімальний номер борта для даного типу ЛА; максимальний номер борта для даного типу ЛА.

Зовнішній вигляд діалогового вікна підсистеми представлений на рис. 3.12.

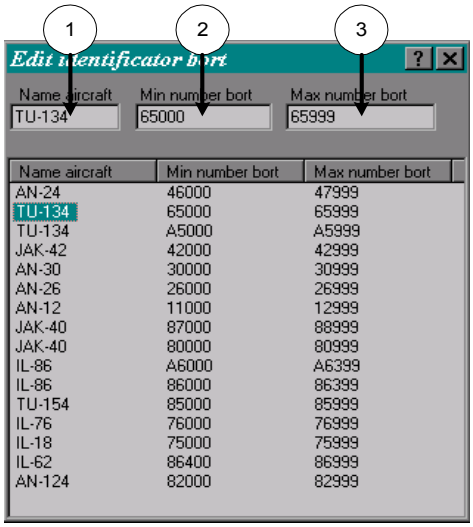

Рис. 3.12. Діалогове вікно підсистеми ідентифікації бортів

1 – поле редагування типу ЛА;

2 – поле редагування мінімального значення борта для даного типу ЛА;

3 – поле редагування максимального значення.

3.3.7 Склад і особливості функціонування управляючої підсистеми

До складу управляючої підсистеми входить ряд програм, що реалізують користувальницький інтерфейс у системі автоматизації сертифікаційних випробувань програмного забезпечення контролю польоту. Програми призначені для полегшення взаємодії користувача із системою обробки ПІ, а також для реалізації більш зручної форми відображення інформації з використанням багатовіконного інтерфейсу і системи багаторівневих меню.

Управляюча підсистема запускається на виконання по команді, введеної з клавіатури. Запуск може виконуватися як без параметрів (із наступним уведенням їх по запиті з програми), так і з зазначенням деяких із них у

командному рядку. Вхідні дані для програми містяться у файлі циклограми. Даний файл містить робочу інформацію про структуру копії польоту даного типу БР і даного типу ЛА, необхідну для виконання подальшої обробки ПІ. Циклограма може бути введена по запиті з програми.

3.3.8 Робота управляючої підсистеми і головне меню

Після запуску системи на екрані з'являється меню й система відображення. Виклик усіх підсистем здійснюється безпосередньо через головне меню. Головне меню розташоване у верхній частині екрана і містить у собі п'ять пунктів, які реалізують наступні функції: файловий сервіс, сервіс редагування, візуалізація параметрів, установка системних параметрів, установка робочих параметрів.

У пункті меню File містяться команди, що реалізують файловий сервіс. Після натискання кнопки Open здійснюється відкриття файлу, що був заданий у діалозі. Ім'я файлу передається управляючій підсистемі, після чого вона робить ініціалізацію всіх необхідних підсистем. При виборі пункту меню Save здійснюється збереження файлу. При виборі Exit виконується вихід із програми, і якщо були зроблені які-небудь зміни, видається запит на збереження файлу зі змінами.

У пункті Edit реалізуються команди для редагування копії. При виборі пункту Undo здійснюється повернення на крок назад у послідовності виконуваних операцій. Після вибору Cut реалізується збереження виділеної ділянки копії з наступним видаленням цієї ділянки. Пункт Copy запам'ятовує виділену ділянку в буфер, а Paste вставляє ділянку, збережену попередньою командою, у поточну позицію курсору. Пункт меню Delete використовується для видалення виділеної користувачем ділянки, а пункт Insert застосовується для додавання обраної ділянки в поточну позицію курсору (візира).

У меню View реалізовані команди для вибору вікон перегляду. Після вибору пункту Analog викликається діалогове вікно представлення даних про АП, вид якого показаний на рис. 3.9. Після вибору пункту меню Binary

викликається діалогове вікно перегляду даних про РК, вид якого представлений на рис. 3.8. Після вибору пункту Mini Map викликається вікно стислого представлення копії польоту, яке зображене на рис. 3.4.

Меню Options надає інструментарій для внесення змін в управляючі дані. Пункт Identif Bort призначений для виклику редактора, який вносить зміни у файл даних нумерації бортів ЛА. Зовнішній вигляд діалогу представлений на рис. 3.12. Пункт Temporary призначений для введення шляху до тимчасових робочих файлів даних. Пункт Add FDR викликає діалогове вікно додавання аплікації для обробки даних нового типа магнітного бортового накопичувача. Меню Tarirovki призначено для виклику спеціалізованого редактора файлів градуювальних характеристик. Зовнішній вигляд даного діалогу представлений на рис. 3.10. Меню Ciklograma призначено для виклику спеціалізованого редактора файлу циклограм. Зовнішній вигляд даного діалогу представлений на рис. 3.11.

Меню Select реалізує інтерфейс вибору користувачем АП і РК. Пункт Analog призначений для виклику діалогового вікна вибору АП із спеціалізованими характеристиками. Зовнішній вигляд даного вікна представлений на рис. 3.5 (підсистема робочих профілів). Меню Binary призначено для виклику діалогового вікна вибору РК із спеціалізованими характеристиками. Крім того, із меню Select можна здійснити виклик підсистеми тестування логіки алгоритмів контролю, а також підсистеми обчислення вірогідності подій контролю.

3.3.9 Робота з графічним середовищем

Після відкриття всіх потрібних для роботи файлів, і вибору всіх бажаних параметрів для редагування, користувач може вибрати ділянку для редагування. У залежності від обраної кількості кадрів викликаються різні редактори (алгоритм показаний на рис. 3.13). Якщо обраний один кадр, то викликається спрощений редактор, показаний на рис. 3.7. У випадку вибору

декількох кадрів інформації викликається редактор, що представлений на рис. 3.6.

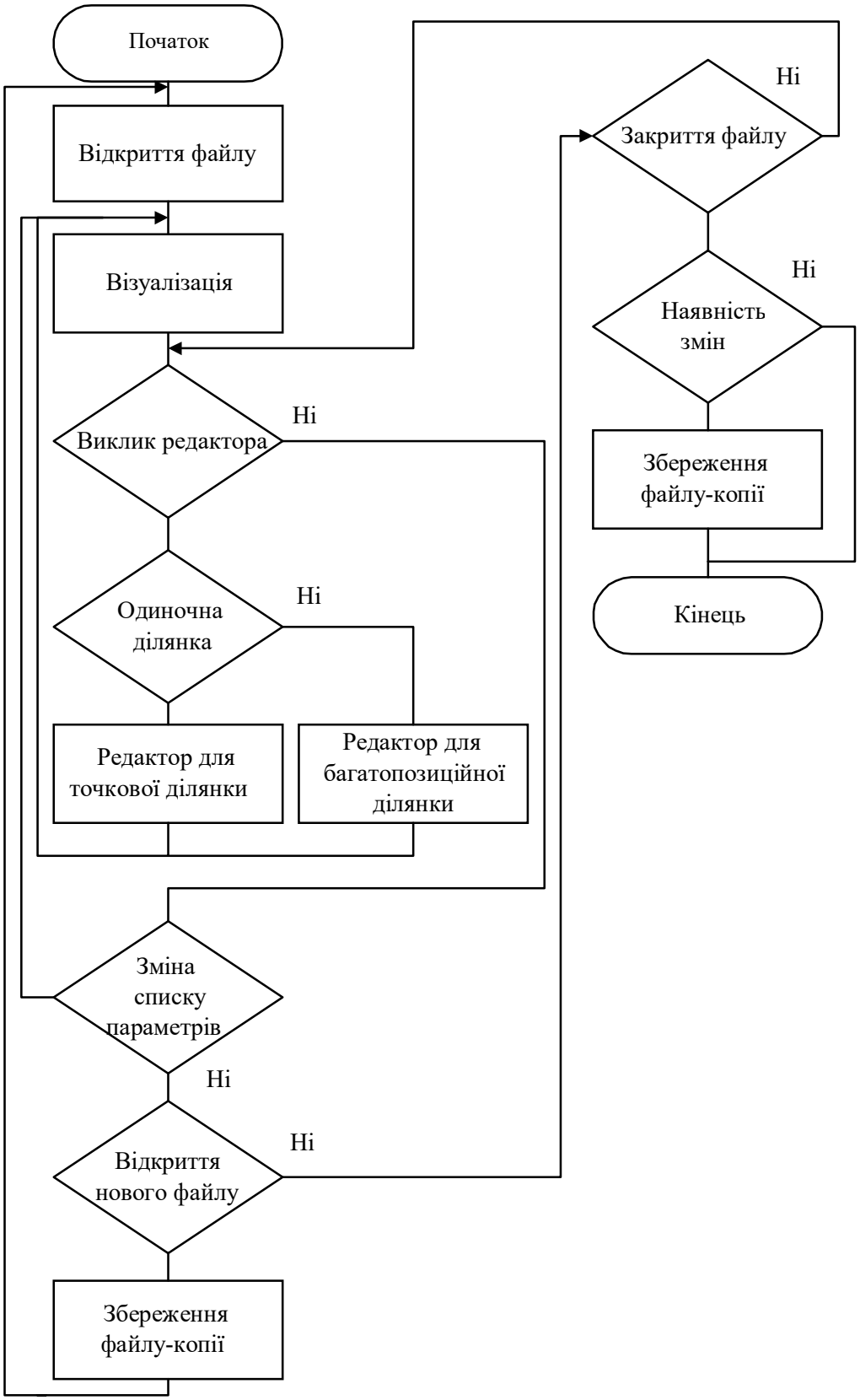

Рис. 3.13. Алгоритм роботи управляючої підсистеми

Висновки до розділу

В даному розділі обґрунтовано функціональну організацію та досліджено аспекти побудови програмної системи, яка забезпечує застосування засобів автоматизації для розрахунку фактичних показників якості й прийняття рішення про відповідність вимогам при сертифікаційних випробуваннях. Описано структуру і комплексу генерації ТНД, який дає можливість автоматизувати створення ТНД для забезпечення процесу випробувань.

ВИСНОВКИ

В магістерській роботі описано моделі та методи забезпечення якості програмних рішень для автоматизованих систем, а також концепцію побудови сертифікаційної моделі якості програмного забезпечення, яка полягає у формуванні вимог до програмного забезпечення із подальшою їх класифікацією згідно характеристик якості стандарту ISO/IEC. Сертифікаційна модель включає сукупність вибраних характеристик, критеріїв відповідності та метрик, що дозволяє отримати уніфіковані процедури оцінки атрибутів якості програмного забезпечення.

Досліджено технологію оцінювання характеристик якості програмного забезпечення, яка містить методи, алгоритми та програмне забезпечення. Технологія дає можливість застосувати засоби автоматизації створення технічної документації та розрахунку фактичних значень показників якості, що спрощує випробування і значно скорочує їх терміни.

На основі аналізу області застосування програмного забезпечення обґрунтовано вибір з існуючих методів тестування тих методів, які забезпечують виявлення невідповідностей програмного забезпечення вимогам у модулях програмного забезпечення автоматизованих систем керування.

Реалізовано прототип системм випробувань програмного забезпечення, обґрунтовано функціональну організацію та досліджено аспекти побудови програмної системи, яка забезпечує застосування засобів автоматизації для розрахунку фактичних показників якості й прийняття рішення про відповідність вимогам при сертифікаційних випробуваннях.

Описано структуру і програмне забезпечення комплексу генерації тестового набору даних, який дає можливість автоматизувати створення даних для забезпечення процесу визначення якості ПЗ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 14598-2. Software engineering – Product evaluation – Part 2: Planning and management, 2000. – 12 p.
2. Voas J., Miller K. Software Testability:The New Verification // IEEE Software. – May-June 1995, pp.17–28.
3. Voas J.,«Software Certification Laboratories: To Be or Not to Be Liable?» // Crosstalk, Apr. 1998, pp. 21-23.
4. Бабенко Л.П., Лаврищева К.М. Основи програмної інженерії. Навч. посіб. – К.: Т-во "Знання", КОО, 2001. – 269 с.
5. Томашевський В.М. Імітаційне моделювання систем і процесів : Навч.посібник. –К.: ІСДО, 1994. –124 с.
6. Томашевський В.М., Жданова О.Г., Жолдаков О.О. Вирішення практичних завдань методами комп'ютерного моделювання : Навч.посіб.–К.: "Корнійчук", 2001.–268 с.
7. ДСТУ 2462–94. Сертифікація. Основні поняття. Терміни та визначення. – Чинний від 01.01.95. –К.: Держстандарт України, 1994. – 27 с.
8. ДСТУ 2851–94. Програмні засоби ЕОМ. Документування результатів випробувань. – Чинний від 01.01.96. –К.: Держстандарт України, 1994. – 12 с.
9. ДСТУ 2853–94. Програмні засоби ЕОМ. Підготовлення і проведення випробувань. – Чинний від 01.01.96. –К.: Держстандарт України, 1994. – 17 с.
10. Харченко О.Г. Система автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів // Вісник Черкаського державного технологічного університету. – 2003. – №3. –С.24–30.
11. ДСТУ ISO 9000-3-98. Стандарти з управління якістю та забезпечення якості. Частина 3. Настанови щодо застосування ДСТУ 9001-95 під час розроблення, постачання та супроводження програмних засобів. –К.:Держстандарт України, 1998.
12. IEEE Std 730-1998. IEEE Standard for Software Quality Assurance Plans, 1998. – 21 p.
13. ISO/IEC 9126. Information Technology – Software product evaluation – Quality characteristics and guidelines for their use, 1991. – 14 p.
14. ISO/IEC 14598-1. Information Technology – Software product evaluation – Part 1: General overview, 1999. – 20 p.

15. Шрюфер Е. Обробка сигналів: Цифрова обробка дискретизованих сигналів. За ред. проф. В.П.Бабака. –К.: Либідь, 1992. – 296 с.
16. Bohm, Corrado; and Giuseppe Jacopini (May 1966). "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". Communications of the ACM 9 (5): 366–371. doi:10.1145/355592.365646
17. Dijkstra, E. W. (Aug 1972). "The Humble Programmer". Communications of the ACM 15 (10): 859–866. doi:10.1145/355604.361591. http://www.cs.utexas.edu/~EWD/transcriptions/EWD03xx/EWD340.html. (EWD340) PDF, 1972 ACM Turing Award lecture
18. Dijkstra, E.W., "Structured Programming," Software Engineering Techniques, Buxton, J.N., and Randell, B., eds. Brussels, Belgium, NATO Science Committee, 1969.
19. B. Meyer, Object-Oriented Software Construction, second ed., Prentice Hall, 1997, Chap. 6, 10, 11.
20. Guide to the Software Engineering Body of Knowledge (SWEBOK). CHAPTER 4. SOFTWARE CONSTRUCTION. http://www.computer.org/portal/web/swebok/html/ch4K. Beck, Test-Driven Development: By Example, Addison-Wesley, 2002.
21. McCabe : Complexity Measure, IEEE Transacions on Software Engineering, Volume 2, No 4, pp 308-320, December 1976
22. M. Fowler and al., Refactoring: Improving the Design of Existing Code, Addison-Wesley, 2002.
23. Russell Gold, Thomas Hammell, Tom Snyder. Test Driven Development: A J2EE Example.- Apress, 2005.- 296 pages
24. Бабенко Л. П. Основи програмної інженерії: навч.посіб. для студ. вищ. навч. закл. Київ: Знання, 2001. 269 с.
25. Винничук Р. О. Особливості розвитку ІТ-ринку в Україні: стан та тенденції. Вісник Національного університету «Львівська політехніка». Серія «Логістика». 2015. № 833. С. 3-8.
26. Вовк І. В. Аналіз інцидентів, спричинених помилками програмного забезпечення. Інтелектуальні технології в системному програмуванні:

матеріали Всеукр. наук.-практ. конф. молодих вчених та студентів. - Хмельницький, 18-19 квітня 2013 р.

27. Говорущенко Т. О. Дослідження відомих моделей оцінювання характеристик програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2013. № 1. С. 117-121.
28. Говорущенко Т. О. Дослідження результатів виявлення помилок програмного забезпечення на різних етапах життєвого циклу. Системний аналіз та інформаційні технології : тези доповідей міжнар. наук.-техн. конф. (Київ, 27- травня 2013 р.). Київ, 2013. С. 411-412.
29. Козак М. В. Реалізація та дослідження нейромережної складової методу оцінювання та прогнозування якості програмного забезпечення. Науковий вісник Чернівецького національного університету. Серія «Комп'ютерні системи та компоненти». 2011. Т. 2. Вип. 1. - 19-26
30. Маєвський Д. А. Теоретичні та прикладні основи забезпечення якості динамічних інформаційних систем: дис. ... доктора техн. наук: 05.13.06. Одеса, 2013. 440 с.
31. Онищук О. С. Оцінювання результатів проектування та прогнозування характеристик якості програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2011. № 2. С. 168-178.
32. Павлова О. О. Метод оцінювання достатності інформації для визначення якості програмного забезпечення на основі зваженої онтології. Вісник Хмельницького національного університету. Серія «Технічні науки». 2016. № 5. С. 146-156.
33. Пиріг С. О. Інформаційні технології та їх використання на підприємствах України. Економічний форум. 2014. № 3. С. 190–195.
34. Питлик Є. В. Визначення ефективності метрик якості на етапі проектування програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2012. № 2. С. 149-155.
35. Поморова О. В. Дослідження засобів оцінки якості на різних етапах життєвого циклу програмного забезпечення. Вісник Національного

університету «Львівська політехніка». Серія «Комп'ютерні системи та мережі». 2011. № 717. С. 141-146.

36. Поморова О. В. Моделювання процесу оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2017. № 6. С. 70-80.
37. Поморова О. В. Проблеми галузі забезпечення якості програмних продуктів. Інтелектуальні технології в системному програмуванні: матеріали Всеукр. наук.-практ. конф. молодих вчених та студентів (Хмельницький, 23-25 квітня 2014 р.). Хмельницький, 2014. С. 390-391.
38. Поморова О. В., Говорущенко Т. О. Дослідження характеристик навчальної вибірки для штучної нейронної мережі оцінювання якості програмного забезпечення. Поступ в науку. Збірник наукових праць Бучацького інституту менеджменту і аудиту. 2012. № 8. С. 147-152.
39. Сеньківський В. М. Синтез моделей пріоритетного впливу факторів на якість процесу створення програмного забезпечення мобільних пристроїв. Поліграфія і видавнича справа. 2016. №1. С. 67-78.
40. Тарасек С. Я. Аналіз та опрацювання метрик якості програмного забезпечення на етапі проектування. Вісник Хмельницького національного університету. Серія «Технічні науки». 2010. №1. С. 54-63.
41. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. Вісник Тернопільського державного технічного університету. 2009. Т. 14. № 1. С. 201-207.
42. Яковина В.С. Якість програмного забезпечення. Інженерія програмного забезпечення. 2010. № 2. С. 24-29.
43. Foidl H., Felderer M. Integrating software quality models into risk-based testing // Software Quality Journal. –V 26. -2018. – P. 809 – 847.
44. Jacob P.M., Mani P. A framework for evaluating performance of software testing tools // International Journal of Scientific and Technology Research. –V. 9. – Issue 2. – 2020. P. 2175–2180.

ДОДАТКИ

Додаток А

Фрагмент програми пошуку події контролю на мові С++

```
#include  <owl.h>
....
unsigned int  tic=0;  // поточний час у тиках від початку копії (1 тик = 0.5
сек.)
unsigned short  cadr[128], *pcadr=&cadr[0];  // кадр копії (синхронізований  з
tic)
….   // декларації
main ( int argc, char *argv[] )
{ …. // декларації
 class  gt_a_rud   // клас  ГТ_αруд – готовність РКД (РУД)
 {                          // алгоритм контролю 2-го класу
  private:
   unsigned short  z_gt_a_rud; // результат роботи автоматної моделі
готовності(0 чи 1)
   unsigned short  alpha1,alpha2,alpha3,n1,n2,n3; //вхідні сигнали (значення
предикатів)
   unsigned short  q1_gt_a_rud; // стан q(t+1), лічильник
   unsigned short  q_gt_a_rud=0; // внутрішній стан автомата q(t)
   unsigned short  z0_gt_a_rud; // результат виклику логічної функції ГТ_αруд
   unsigned int  ctic;  // тик (точка) останнього виклику
   unsigned int  d_tau; // Δτ (дельта тау) – інтервал часу функціонування
  public:
   void gt_a_rud() {z_gt_a_rud=0;q1_gt_a_rud=0;z0_gt_a_rud=0; ctic=0;} //
конструктор
   void ~gt_a_rud();  // деструктор

  unsigned short  fz_gt_a_rud ( unsigned int  p_tic, unsigned short  *p_cadr);
  { alpha1 = predicate(1,19,2,100,1,-1,p_tic,p_cadr);   // функція predicate
повинна бути //описана в загальній частині програми і повинна бути доступною;
її параметри //мають наступні значення: 1) тип (0 - РК, 1 - АП, 2 - Δτ); 2)
номер каналу реєстрації; //3) тип  обмеження  (0 - ∃, 1 - ′, 2 - /, 3 - ≤);   4)
фізичне значення обмеження; //5) вірогідність (0 - не визначати, 1 - визначати,
і якщо вірогідність низька, то //функція повертає -1); 6) якщо тип РК, то задає
номер біта в каналі,інакше поле = -1; //7)номер тику; 8)адреса кадру в копії; В
цілому функція predicate обчислює простий //одномісний предикат і повертає 1,
якщо предикат дорівнює 1.
      alpha2 = predicate(1, 23, 2, 100, 1, -1, p_tic, p_cadr);
      alpha3 = predicate(1 ,31, 2, 100, 1, -1, p_tic, p_cadr);
```

```
      n1 = predicate(1, 25, 2, 85, 1, -1, p_tic, p_cadr);
      n2 = predicate(1, 35, 2, 85, 1, -1, p_tic, p_cadr);
      n3 = predicate(1, 43, 2, 85, 1, -1, p_tic, p_cadr);
      z0_gt_a_rud = fz0_(); // визначення логічної функції автомата
      if(ctic!=0)
         if ( (p_tic-ctic>1) || (p_tic-ctic<=0) ) { d_tau=0; q_gt_a_rud=0;
q1_gt_a_rud=0;}
      d_tau = fq_gt_a_rud(); // стан автомата акумулює інтервал функціонування
      z_gt_a_rud = z0_gt_a_rud . predicate(2, d_tau, 2, 8, 0, -1, p_tic,
p_cadr);
// функція автомату згідно канонічного рівняння z(t) = z0(t) . (q(t)/8)
    return( z_gt_a_rud);  }

 unsigned short  fz0_( void)   // визначення логічної функції ГТ_αруд
 { return( alpha1&&alpha2&&alpha3&&n1&&n2&&n3);  }

 unsigned short  fq_gt_a_rud ( void)  // функція стану автомата q(t)
 {
    if (z0_gt_a_rud  = = 1)  {
        q1_gt_a_rud++; q_gt_a_rud = z0_gt_a_rud . (q1_gt_a_rud - 1); }
    else { q1_gt_a_rud=0; q_gt_a_rud=0; }
    return(q_gt_a_rud);   }

friend unsigned short predicate (unsigned short, unsigned int, unsigned short,
unsigned int, unsigned short, unsigned short, unsigned int, unsigned short *);
}   // кінець описання класу ГТ_αруд

class p_rul { // признак рулювання – алгоритм контролю 1-го класу
 private:
    unsigned short  z_p_rul;  // функція автомату
    unsigned short  n11, n22, n33; // вхідні сигнали оборотів роторів двигунів
    unsigned short  gt_a; // результат виклику функції gt_a_rud (готовність
РКД)
    unsigned short  q_p_rul, q1_p_rul; // внутрішні стани автомата
    unsigned short  z0_p_rul; // логічна функція автомата p_rul
  p_rul() {q_p_rul=q1_p_rul=0; z0_p_rul=z_prul=0;}; ~p_rul(){}; //конструктор

public:
friend unsigned short predicate (unsigned short, unsigned int, unsigned short,
unsigned int, unsigned short, unsigned short, unsigned int, unsigned short *);

   unsigned short int fz_p_rul ( unsigned int p_tic, unsigned short * p_cadr)
```

```
 {  g_ta=fz_gt_a_rud(p_tic, p_cadr); // виклик готовності РКД (РУД)
     if( gt_a == 0) g_ta =1; else g_ta = 0; // заперечення готовності
     n11=predicate(1, 21, 1, 85, 1, -1, p_tic, p_cadr); //21,29,33- обороти
роторів двигунів
     n22=predicate(1, 29, 1, 85, 1, -1, p_tic, p_cadr);
     n33=predicate(1, 33, 1, 85, 1, -1, p_tic, p_cadr);
     z0_p_rul = fz0_p_rul(); q_p_rul = fq_p_rul(); q1_p_rul = q_p_rul;
     z_p_rul = z0_p_rul;
     return( z_p_rul);   }

 unsigned short fz0_p_rul( void)
     { z0_p_rul = g_ta&&n11&&n22&&n33; return( z0_p_rul); }

 unsigned short fq_p_rul( void) { q_p_rul = z0_p_rul; return( q_p_rul); }
}  // кінець описання класу p_rul

class  s_012  {  // подія (алгоритм) контролю S012
                        // алгоритм контролю 2-го класу
private:
   unsigned short  z_s_012; // результат роботи автоматної моделі події
контролю
   unsigned short  i; // разова команда включення протиобледеніння
   unsigned short  q1_s_012; // стан q(t+1), лічильник
   unsigned short  q_s_012 = 0; // внутрішній стан автомата q(t)
   unsigned short  z0_s_012; // результат виклику логічної функції S012
   unsigned int  ctic;  // тик (точка) останнього виклику
   unsigned int  d_tau; // Δτ (дельта тау) – інтервал часу функціонування
public:
   void  s_012() {z_s_012=0; q1_s_012=0; z0_s_012=0; ctic=0;} // конструктор
   void ~s_012();  // деструктор
  unsigned short  fz_s_012( unsigned int p_tic, unsigned short * p_cadr)
  {
     i = predicate(0, 15, 0, -1, -1, 4, p_tic, p_cadr); z0_s_012 = fz0_s_012();
     if(ctic!=0)
       if ( (p_tic-ctic>1) || (p_tic-ctic<=0) ) { d_tau=0; q_gt_a_rud=0;
q1_gt_a_rud=0;}
     d_tau=fq_s_012();  // інтервал функціонування
     z_s_012 = z0_s_012 . predicate(2, d_tau, 2, 1230, 0, -1, p_tic, p_cadr);
    return( z_s_012);
  }

  unsigned short  fz0_s_012( void)  {
```

```
     z0_s_012 = fz_p_rul(p_tic, p_cadr) . i;
    return( z0_s_012);  }

  unsigned short fq_s_012( void)  // функція стану автомата q(t)
  {
    if (z0_s_012  = = 1)  {
        q1_s_012++; q_s_012 = z0_s_012 . (q1_s_012 - 1); }
    else { q1_s_012 = 0; q_s_012 = 0; }
    return(q_s_012);   }
 }   // кінець описання класу s_012
....
 gt_a_rud  g;  // екземпляр готовності
 p_rul  p;       // екземпляр рулювання
 s_012  s12;  // екземпляр алгоритму контролю
....// оператори
 while ( (cadr=read_cadr()) != EOF ) // поки не кінець параметричної копії
{
   tic++;
….// оператори
   if( s12::fz_s_012( tic, cadr) = = 1)
       break;  // вихід із циклу: контрольована подія знайдена *****
....// оператори
 }  //  кінець циклу while
....// оператори
}   // кінець програми main()
```