УДК 004.4:006.015.5

І.Е. РАЙЧЕВ, О.Г. ХАРЧЕНКО

КОНСТРУЮВАННЯ ПРОГРАМ СТВОРЕННЯ ТЕСТОВИХ НАБОРІВ ДАНИХ НА БАЗІ АВТОМАТНИХ МОДЕЛЕЙ

Abstract: *In the article the questions of automation of creating test data sets are considered. The technology of creating programs of generation of test data sets which is based on construction of automatic models is offered. The obtained test data sets allow to automate detection of events of the monitoring that raises efficiency of tests, reduces labour input of certification and increases reliability of their results.*
Key words: *test data sets, automatic models, certification of program systems, automation of testing.*

Анотація: *У статті розглянуті питання автоматизації створення тестових наборів даних (ТНД). Для цього запропонована технологія створення програм генерації ТНД, яка базується на побудові автоматних моделей. Отримані ТНД дозволяють автоматизувати виявлення подій контролю, що дає можливість підвищити ефективність випробувань, зменшити трудомісткість сертифікації та збільшити достовірність її результатів.*
Ключові слова: *тестові набори даних, автоматні моделі, сертифікація програмних систем, автоматизація тестування.*

Аннотация: *В статье рассмотрены вопросы автоматизации создания тестовых наборов данных (ТНД). Предложена технология создания программ генерации ТНД, которая базируется на построении автоматных моделей. Полученные ТНД позволяют автоматизировать выявление событий контроля, что повышает эффективность испытаний, уменьшает трудоемкость сертификации и увеличивает достоверность ее результатов.*
Ключевые слова: *тестовые наборы данных, автоматные модели, сертификация программных систем, автоматизация тестирования.*

1. Вступ

При сертифікації програмного забезпечення автоматизованих систем контролю (ПЗ АСК) виникає проблема створення тестових наборів даних, які б покрили всі можливі маршрути виконання програм, що реалізують алгоритми контролю [1]. Це пов'язано з тим, що крім самих алгоритмів контролю, які містять десятки предикатів і логічних функцій, в ПЗ АСК реалізуються також алгоритми пошуку контрольованих ситуацій, які є не менш складними логічними виразами. Зважаючи на той факт, що для кожного об'єкта контролю таких алгоритмів декілька сотень, створення надійного й ефективного програмного забезпечення контролю є складною задачею. ПЗ АСК, як правило, створюється із застосуванням підходу прямого кодування алгоритмів контролю [2], що створює високу ймовірність наявності невиявлених помилок [3] і ускладнює перевірку правильності програм. Високий ступінь вкладеності конструкторів розгалуження призводить до труднощів при трасуванні та побудові вичерпного набору тестів, що спричиняє наявність помилок у програмах, які реалізують алгоритми (події) контролю.

Ця робота присвячена розробці методів автоматизації створення ТНД при сертифікаційних випробуваннях ПЗ АСК.

2. Застосування теорії автоматів для програмування алгоритмів контролю та побудови тестових наборів даних

Одним із шляхів вирішення проблеми покриття графа передач управління модулів ПЗ АСК при його тестуванні є використання формальних методів, які дають можливість провести трасування та всеосяжний аналіз логіки програм контролю й застосувати засоби автоматизації програмування та прогресивні технології програмування (Р-технологія, об'єктно-орієнтоване проектування, автоматне

програмування, UML та ін.) [4, 5]. Оскільки вхідна інформація для алгоритмів контролю є дискретною, а вихідна – скінченною множиною дискретних подій, то для моделювання (імітації) таких алгоритмів авторами був застосований математичний апарат дискретних скінченних автоматів [6, 7]. У зв'язку з відсутністю в алгоритмах контролю кванторів існування й загальності значно полегшується застосування теорії скінченних автоматів при аналізі й синтезі відповідних автоматних моделей. Предикати в алгоритмах контролю мають вид нерівностей і зводяться до одномісних. Наприклад, для літальних апаратів: $V_{пр}$/410, $\alpha_{руд}$'110, $H_{Г}$/15, $H_{б}$/7120 і т.п. ($V_{пр}$ – швидкість приладова, $\alpha_{руд}$ – кут відхилення ручки керування (управління) двигуном (РКД або РУД), $H_{Г}$ – висота геометрична, $H_{б}$ – висота барометрична).

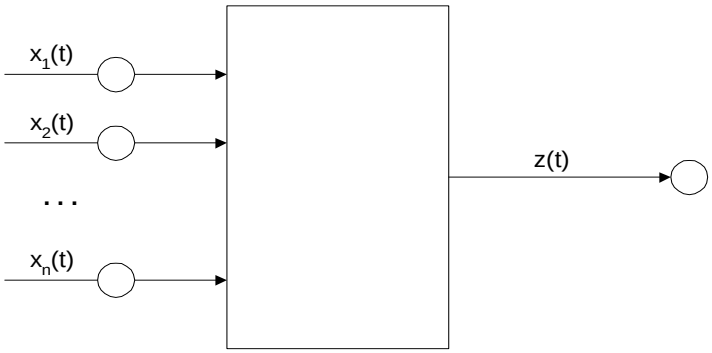

Рис. 1 Функціональна схема автомата з одним виходом

Автомати, які реалізують предикати такого типу, можна представити у такому вигляді:

На рис. 1 $x_1(t), x_2(t), ..., x_n(t)$ суть функції, які приймають значення 0 або 1 і визначаються через предикати алгоритмів контролю. Наприклад,

$$x_1(t) = V_{пр} / 410,$$

$$x_2(t) = \alpha_{руд}\text{'}110, ..., x_n(t) = H_{б} / 7120.$$

Спростивши логічні функції й предикати, що описують алгоритм контролю, та привівши алгоритм до довершеної кон'юнктивної нормальної форми, побудуємо систему канонічних рівнянь скінченного автомата [8, 9], яка в загальному вигляді може бути представлена рівняннями

$$\left.\begin{aligned} z(t) &= \Phi[x_1(t), x_2(t), ..., x_n(t), q(t)] \\ q(t+1) &= \Psi[x_1(t), x_2(t), ..., x_n(t), q(t)] \\ q(0) &= 0, t = 0,1,2,... \end{aligned}\right\}. \qquad (1)$$

Рівняння (1) і функції Φ і ψ є канонічними рівняннями і функціями над алфавітами X, Z, Q. (X – вхідний алфавіт, Z – вихідний алфавіт, Q – алфавіт станів, $q(t)$ – функція стану автомата). Оскільки на виході маємо один полюс, на якому виникає 1 або 0 (подія відбулася або ні), маємо одну результуючу функцію – $z(t)$.

З метою автоматизації моделювання алгоритмів контролю та генерації ТНД виконано уніфікацію математичного представлення алгоритмів контролю. Для цього була проведена класифікація цих алгоритмів, і на прикладі програмного забезпечення автоматизованих систем контролю польотів (ПЗ АСКП) досліджені та реалізовані алгоритми контролю польотів повітряних суден, що дало можливість синтезувати автоматні моделі, за допомогою яких формулюється й вирішується задача автоматичного пошуку подій контролю за параметричною інформацією, як показано у працях [6, 7]. За цією класифікацією виділено алгоритми контролю 1-го класу, що

інтерпретуються як автомат без пам'яті, і алгоритми контролю 2-го класу, які є автоматами з пам'яттю, де стан автомата акумулює тривалість події.

Розробка автоматних моделей базується на методиці синтезу скінченних автоматів, що містить побудову канонічних рівнянь і логічних схем алгоритмів контролю. Зрештою, за допомогою програмування автоматних моделей, може бути вирішена задача програмного пошуку подій контролю, а також задача побудови методики оптимального тестування множини алгоритмів контролю, що дозволить автоматизувати створення тестових наборів даних, які являються покриттям маршрутів виконання алгоритмів контролю.

Застосування отриманих у [6, 7] автоматних моделей для програмування алгоритмів контролю істотно полегшує конструювання програм генерації ТНД при сертифікаційних випробуваннях ПЗ АСК. Для цього необхідно розробити процедуру побудови програм, які б автоматизували пошук контрольованих подій та небезпечних відхилень у роботі об’єкта контролю і дозволяли генерувати ТНД, що моделюють такі події.

3. Конструювання програм пошуку ситуацій контролю та створення ТНД з використанням автоматних моделей

Оскільки кількість ситуацій при контролі будь-якого об'єкта, як правило, значна (декілька сотень), то автоматизація їхнього тестування (поглиблене тестування й аналіз складових елементів алгоритмів контролю) підвищить ефективність тестування ПЗ АСК у цілому та збільшить надійність їхнього функціонування й безпеку життєдіяльності об'єкта контролю. Для автоматизації тестування використаємо автоматні моделі, побудовані на базі скінченних автоматів. По канонічних рівняннях автоматів можна розробити логічні схеми і побудувати програми, що генерують ТНД і здійснюють пошук ситуацій контролю для оцінки відповідних показників якості. Ефективність автоматного програмування показано в публікаціях [5, 10, 11].

Зауважимо, що при прямому програмуванні складних алгоритмів контролю, які включають десятки предикатів і логічних умов, а також декілька тривалостей у часі, значно зростає ймовірність появи помилок. Цих помилок можна уникнути, використавши нижчевикладені принципи конструювання правильно функціонуючих програм за логічними схемами автоматних моделей контролю.

Алгоритми програм пропонується конструювати відповідно до отриманих канонічних рівнянь і логічних схем автоматних моделей, які представляють відповідні алгоритми контролю. Правильність побудованих програм підтверджується фактом адекватності автоматних моделей як пристроїв синхронної дії алгоритмам контролю, бо автоматні моделі за способом побудови правильно інтерпретують алгоритми контролю [6–9]. Отриманий метод базується на програмуванні канонічних рівнянь автомата. Запрограмувати автоматну модель по події контролю – це значить побудувати канонічні рівняння і логічну схему автомата даної події, а потім записати ці рівняння (алгоритм автомата) мовою програмування. Інакше кажучи, потрібно запрограмувати функцію виходу автомата, функцію його стану та описати структури вхідних і вихідних даних.

Необхідно відзначити, що отримана методика програмування автоматних моделей відрізняється від методології автоматного програмування тим, що вона базується на побудованих наперед канонічних рівняннях і логічних схемах. Це дозволяє значно спростити кодування

автоматних моделей у порівнянні з методологією автоматного програмування [5, 10, 11], де спочатку необхідно здійснити побудову таблиць станів, розробку діаграм і алгоритмів функціонування автомата. Наявність формалізму канонічних рівнянь автомата дає можливість уникнути виконання цих етапів, оскільки алгоритм автомата заданий у його рівняннях і побудованих на їхній основі логічних схемах. Отже, у роботах [6, 7] отримані дійсно корисні результати, оскільки далеко не завжди і не для всяких класів задач вдається побудувати канонічні рівняння автомата.

Кодування алгоритмів контролю виконується за типовою схемою для довільної об'єктно-орієнтованої мови програмування за допомогою розробленої у дійсній роботі методики, яка використовує стандартні шаблони для алгоритмів контролю 1-го і 2-го класів, що значно спрощує написання програм. Отримані програми дозволяють виявляти шукані події контролю і вносити події в копію параметричної інформації, яка відображає роботу об'єкта контролю, здійснюючи тим самим побудову ТНД (п.3).

У даній роботі для програмування автоматних моделей були використані об'єктно-орієнтований підхід і мова програмування C++. Події контролю описані за допомогою класів, елементи кожного з яких містять вид алгоритму, предикати, логічні умови й обмеження, вхідні й вихідні сигнали, а також допускові та довірчі інтервали. Довірчі інтервали отримані по алгоритмах, розроблених у праці [12]. Функції-члени класів описують функцію автомата та функцію його стану і служать для пошуку подій контролю, а дружні функції класів служать для генерації ТНД із метою перевірки встановлення даної події модулями АСКП.

За допомогою автоматних моделей побудовано ПЗ виявлення подій контролю в параметричній інформації та створення мінімально-достатніх ТНД. Мінімальність і достатність розуміється в практичному сенсі – розроблені програми перебирають комбінації логічних умов і тривалостей, які приводять до настання контрольованих подій (або їх відсутності), покриваючи конструкції заданих алгоритмів контролю.

Таке ПЗ може будуватися на базі автоматних моделей, запрограмованих у виді

1) програм, що спроектовані із застосуванням мов SDL або UML;

2) програм на об'єктно-орієнтованих мовах (C++, Object Pascal);

3) програм із використанням Р-технології [4].

Розглянемо приклад конструювання програми по алгоритму контролю пілотування літака Ту-154Б S012: "Перевищення часу безперервного обігріву ППД перед злетом", що записується у виді

$$S012 = П_{рул.} \wedge i_{посппд} \wedge (\Delta\tau_2/615) ,$$

де $\Delta\tau_2$ – відрізок часу більший чи рівний 615 сек. реєстрації (2 точки реєстрації за 1 сек.);

$i_{посппд}$ – РК "Система проти обледеніння і обігріву ППД увімкнена";

$П_{рул.}$ – ознака рулювання, що записується в такий спосіб:

$$П_{рул.} = \wedge ГТ_{\alpha руд1\text{-}3} \wedge [(n_{нд1}' 85) \wedge (n_{нд2}' 85) \wedge (n_{нд3}' 85)] ,$$

где $n_{нд1}, n_{нд2}, n_{нд3}$ – обороти роторів 1-го, 2-го і 3-го двигунів.

$$ГТ_{\alpha руд1\text{-}3}=(\alpha_{руд1}/100)\wedge(\alpha_{руд2}/100)\wedge(\alpha_{руд3}/100)\wedge(n_{нд1}/85)\wedge$$
$$\wedge(n_{нд2}/85)\wedge(n_{нд3}/85)\wedge(\Delta\tau_1/4) ,$$

де $\alpha_{руд1}, \alpha_{руд2}, \alpha_{руд3}$ – положення РУД 1-го, 2-го і 3-го двигунів. Тоді

S012=($\alpha_{руд1}$/100)∧($\alpha_{руд2}$/100)∧($\alpha_{руд3}$/100)∧($n_{нд1}$/85)∧($n_{нд2}$/85)∧($n_{нд3}$/85)∧($\Delta\tau_1$/4) ∧

∧ ($n_{нд1}$′85)∧($n_{нд2}$′85)∧($n_{нд3}$′85) ∧ $i_{посппд}$ ∧ ($\Delta\tau_2$/615).

Для події контролю маємо такий набір предикатів:

$\alpha_1 = \alpha_{руд1}/100$, $\alpha_2 = \alpha_{руд2}/100$, $\alpha_3 = \alpha_{руд3}/100$, $n_1 = n_{нд1}/85$, $n_2 = n_{нд2}/85$,

$n_3 = n_{нд3}/85$, $n_{11} = n_{нд1}'85$, $n_{22} = n_{нд2}'85$, $n_{33} = n_{нд3}'85$, $i = i_{посппд}$.

Позначимо $z0\_gt\_a\_rud(t) = \alpha_1(t) \wedge \alpha_2(t) \wedge \alpha_3(t) \wedge n_1(t) \wedge n_2(t) \wedge n_3(t)$. Тоді канонічні рівняння для готовності **ГТ$_{\alpha руд1\text{-}3}$** як алгоритму 2-го класу, враховуючи (1), а також праці [6, 7, 13], будуть мати вигляд

$$\left.\begin{array}{l} z\_gt\_a\_rud(t) = z0\_gt\_a\_rud(t) \wedge (q\_gt\_a\_rud(t) \geq 8) \\ q\_gt\_a\_rud(t+1) = z0\_gt\_a\_rud(t) \wedge (q\_gt\_a\_rud(t) + z0\_gt\_a\_rud(t)) \\ q\_gt\_a\_rud(0) = 0, t = 0,1,2,... \end{array}\right\}. \qquad (2)$$

Для обчислення суми в рівнянні стану автомата системи (2) застосовують відомий оператор суматора послідовної дії [9], але це потрібно лише для побудови принципової електричної схеми автомата, а в разі програмування автомата для накопичення тривалості події достатньо використати лічильник.

Позначимо $z0\_p\_rul(t) = \neg z\_gt\_a\_rud(t) \wedge n_{11}(t) \wedge n_{22}(t) \wedge n_{33}(t)$. Тоді канонічні рівняння для ознаки $П_{рул.}$ як алгоритму 1-го класу запишуться у вигляді системи (3):

$$\left.\begin{array}{l} z\_p\_rul(t) = z0\_p\_rul(t) \\ q\_p\_rul(t+1) = z0\_p\_rul(t) \\ q\_p\_rul(0) = 0, t = 0,1,2,... \end{array}\right\}. \qquad (3)$$

Нарешті, позначимо $z0\_S\_012(t) = z\_p\_rul(t) \wedge i$. Тоді канонічні рівняння для алгоритму контролю 2-го класу S012 запишуться у вигляді системи (4) :

$$\left.\begin{array}{l} z\_S\_012(t) = z0\_S\_012(t) \wedge (q\_S\_012(t) \geq 1230) \\ q\_S\_012(t+1) = z0\_S\_012(t) \wedge (q\_S\_012(t) + z0\_S\_012(t)) \\ q\_S\_012(0) = 0, t = 0,1,2,... \end{array}\right\}. \qquad (4)$$

Аналіз події контролю полегшує логічна схема, приведена на рис. 2.

В ній символом ⊕ позначаються суматори станів автоматів, які можна запрограмувати, наприклад, за допомогою лічильника (стан автомата зберігає тривалість контрольованої події). Подібні складні й неосяжні події важко аналізувати навіть експерту. Отримана логічна схема містить композицію двох автоматів 2-го роду ($ГТ_{\alpha руд1\text{-}3}$ і S012) і автомата 1-го роду ($П_{рул.}$, що включає в себе $ГТ_{\alpha руд1\text{-}3}$), канонічні рівняння яких ми отримали за правилами, розробленими у [6, 7]. Маючи канонічні рівняння і логічну схему, конструюємо правильну програму контролю за правилами, викладеними в [13].

Розглянемо цей процес більш докладно. Помітимо, що в даній автоматній моделі реалізовані ознака етапу $П_{рул.}$ і готовність $ГТ_{\alpha руд1\text{-}3}$, за якими один раз побудувавши автоматні моделі та програми, можна багаторазово їх використовувати в різних алгоритмах контролю.

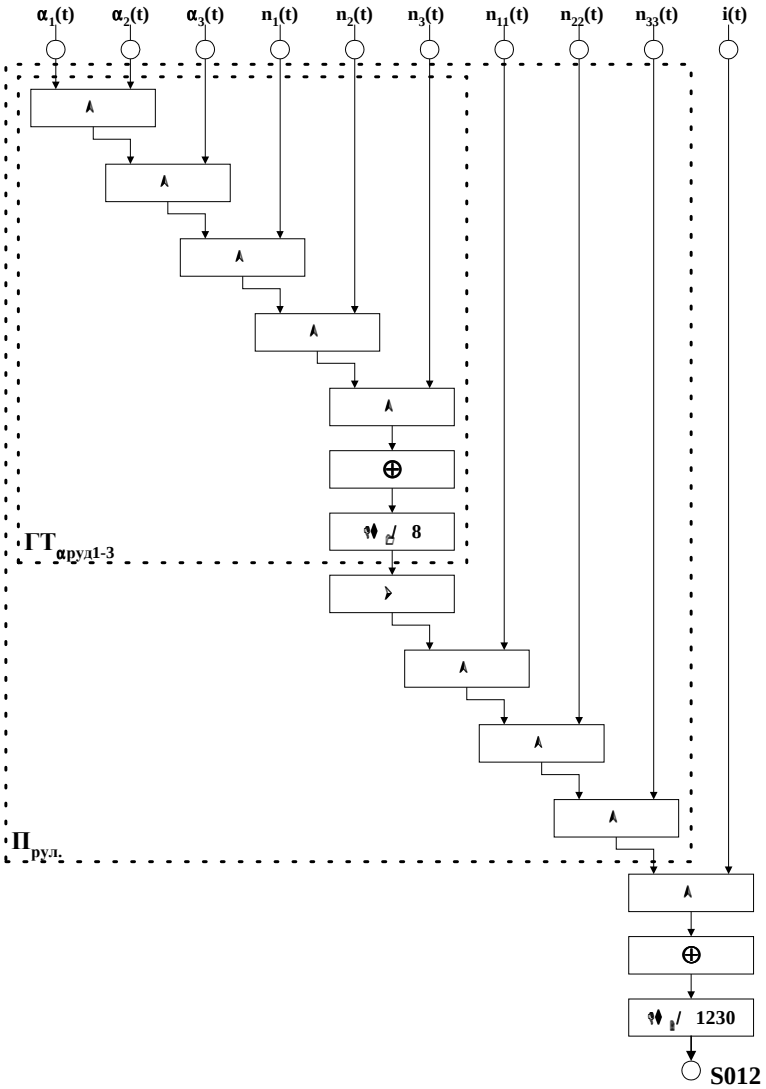

Рис. 2. Логічна схема автоматної моделі по події контролю S012

Якщо ми запрограмували автоматну модель будь-якої події контролю, то це відразу дає нам інструмент пошуку (виявлення) цієї події. Для того ж, щоб вносити в параметричну копію функціонування об'єкта контролю деякі події контролю (тобто створювати різні ТНД), необхідно внести в клас, який описує цю подію, функцію (наприклад, дружню функцію класу), що буде під час проходження програми по параметричній копії збільшувати (чи зменшувати) необхідні параметри (наприклад, висоту, швидкість, оберти двигунів і т.п.), змінюючи їх значення до досягнення ними контрольованої граничної величини з довірчого інтервалу. Таким способом можна будувати різноманітні ТНД, у яких будуть перебиратися необхідні комбінації предикатів, логічних умов і тривалостей, включаючи готовності й ознаки етапів польоту.

За запитом експерта може здійснюватися задане покриття логіки з набору алгоритмів контролю, тобто включення в тестовий набір даних необхідних подій контролю, а також індикація предикатів і логічних умов у складі кожної події, для яких потрібно створити тест.

Таким чином, у противагу програмам контролю, які можуть містити логічні помилки, бо вони побудовані способом прямого програмування, шляхом автоматного програмування отримано правильні програми.

4. Методика програмування автоматних моделей контролю і приклад типового конструювання програм із використанням шаблонів проектування

Особливістю експлуатації критичних систем є те, що у процесі їх функціонування реєструється параметрична інформація, яка дозволяє відтворити стан об'єкта контролю та управляючі впливи. Нормативними документами по експлуатації передбачені обмеження на параметри функціонування об'єкта контролю на певних етапах (режимах) і в певних ситуаціях. ПЗ АСК, як правило, призначене для пошуку подій виходу за обмеження контрольованих параметрів по зареєстрованій інформації

при розслідуванні нештатних ситуацій, аварій та катастроф. Тому для перевірки правильності функціонування ПЗ АСК необхідно створювати ТНД, що можна виконати шляхом внесення відповідних коректив в зареєстровану інформацію.

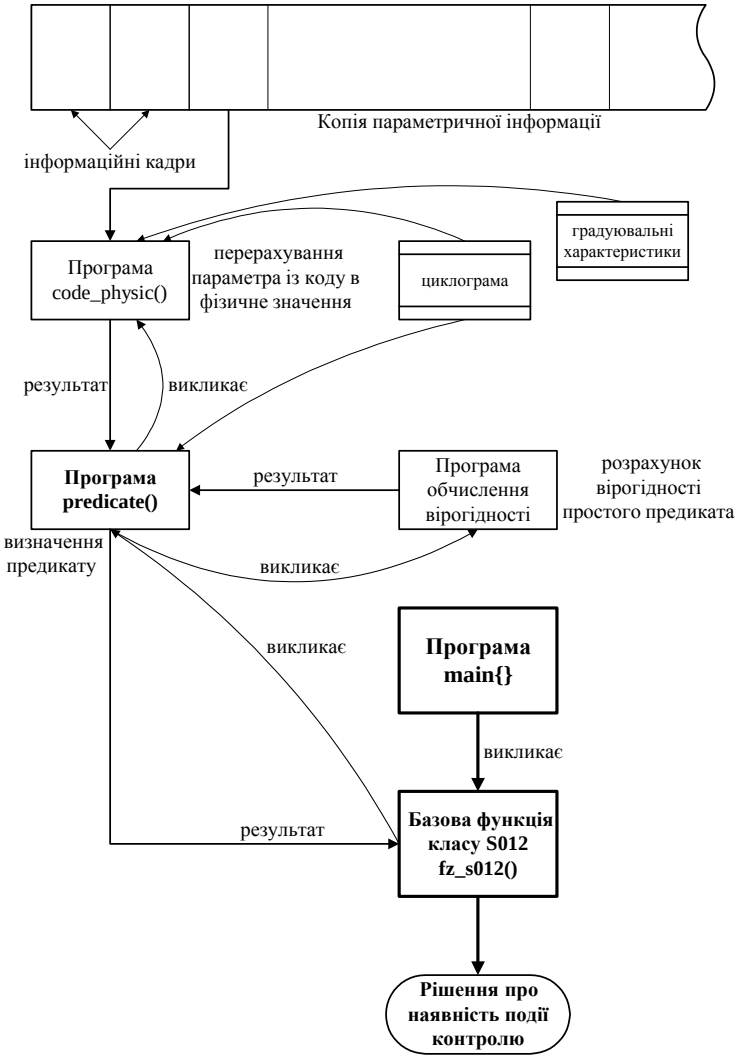

Рис. 3. Технологічна схема визначення подій контролю на прикладі події S012

Пошук та обробка контрольованих подій виконується за інформацією, записаною у відповідності з дискретним часовим рядом. У випадку знаходження події, функція автомата стає рівною одиниці і обнуляється в наступній точці часу.

Як ілюстрацію продемонструємо процес побудови програми на прикладі події контролю S012. Визначимо елементарний базовий клас для $ГТ_{αруд}$ (gt_a_rud), а також класи для $П_{рул}$ (p_rul) і S012 (s_012). Клас p_rul використовує структури даних і викликає базову функцію з gt_a_rud, а клас s_012 використовує функцію класу p_rul і дані з обох підпорядкованих класів. Тому останні два класи можна вважати похідними. На відміну від них елементарні (неподільні) класи не містять у собі елементів старших класів і не звертаються до елементів функцій старших класів.

Predicate() є дружньою функцією всіх базових класів. Вона обчислює предикат по набору вхідних даних автоматної моделі як пристрою синхронної переробки інформації. Будемо викликати цю функцію з головної функції класу – fz_(), яка здійснює пошук події контролю і повертає головне дане класу – z_. Вводиться також додаткова функцію класу cz_(), яка буде автоматично вносити контрольовану подію в тестову копію параметричної інформації, здійснюючи тим самим побудову ТНД на заданому інтервалі. Для цієї функції задається точка часу і максимальне (чи мінімальне) значення параметра, що повинне бути досягнуте в цій точці і яке перевищує (або є нижчим) значення обмеження. Побудова події виконується шляхом проведення гладкої опуклої вгору (чи вниз) кубічної параболи в разі сплайн-інтерполяції, або полінома ступеня не більшої $(n-1)$, де n – кількість точок інтерполяції.

Технологічна схема пошуку подій контролю показана на рис. 3. Подібним чином будуються будь-які програми визначення подій по параметричній інформації.

Приклад побудованої контрольованої події показаний на рис. 4.

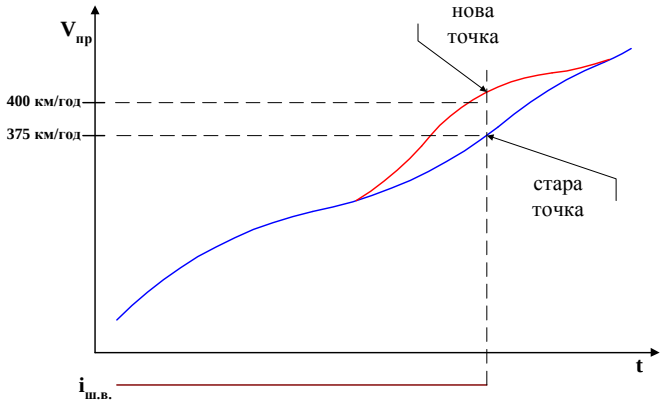

Рис. 4. Внесення події контролю S068: “Перевищення швидкості при прибиранні шасі (400 км/год)” в копію параметричної інформації з метою створення ТНД

Методика побудови програм по автоматних моделях дає можливість використовувати при цьому шаблони. Маються 2 шаблони: 1) шаблон програми по алгоритму контролю 1-го класу; 2) шаблон програми по алгоритму контролю 2-го класу. Для побудови у програмі нових класів, що представляють будь-які інші алгоритми контролю, скористаємося цими шаблонами:

1) заміняємо в обраному шаблоні ідентифікатори подій, готовностей і ознак на потрібні імена;

2) змінюємо предикати (при звертанні до функції predicate() змінюємо тип обмеження, значення обмеження, тривалість, розмірність і т.д., у залежності від виду предиката);

3) змінюємо загальний вид функції автомата на необхідний (формула, що залежить від предикатів і обчислюється шляхом застосування правил із п. 2).

Запропонованим способом можна легко побудувати набір класів, що покривають множину заданих алгоритмів контролю об'єкта. Побудовані класи здійснюють автоматичний пошук подій контролю в параметричній інформаційній копії. Тому значно спрощується написання й налагодження програм контролю, а також програм побудови ТНД, що містять події контролю (внесення подій у параметричну копію). Фрагмент програми пошуку події контролю S012 на мові C++ буде виглядати так:

```
#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;}  void ~gt_a_rud(); // конструктор і 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 має 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. predicate обчислює простий одномісний предикат і повертає 1, якщо предикат дорівнює true.
     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 автомата згідно з канонічним рівнянням 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 автомата
    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
public:
  p_rul() {q_p_rul=q1_p_rul=0; z0_p_rul=z_prul=0;};  ~p_rul(){};  // конструктор і деструктор
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 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 проти обледеніння
   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;} ~p_rul(){};  // конструктор і деструктор
  unsigned short  fz_s_012( unsigned int p_tic, unsigned short * p_cadr)  // функція події S012
  {  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 S012
....// 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 *****
 }  //  кінець циклу while
....// оператори
}   // кінець програми main()
```

5. Висновки

Таким чином, отримано результати:

1) На базі автоматних моделей розроблено методику тестування алгоритмів контролю, що полягає в автоматичній генерації ТНД, які є мінімально-достатніми покриттями маршрутів виконання програм, реалізуючих події контролю. Розроблено рекомендації з практичного відстеження подій контролю при програмуванні модулів контролю, що знижує кількість помилок при написанні та налагодженні програм.

2) З використанням об'єктно-орієнтованого підходу по канонічних рівняннях і логічних схемах автоматних моделей контролю розроблено методику конструювання правильно функціонуючих програм пошуку контрольованих подій. Отримані програми дозволяють автоматизувати виявлення цих подій і генерувати ТНД, що дає інструмент визначення фактичних показників якості виявлення небезпечних відхилень для ПЗ АСК.

3) На основі отриманих методів побудовано підсистему тестування логіки алгоритмів контролю, яка входить у комплекс проектування та генерації ТНД системи автоматизації сертифікаційних випробувань і створює події контролю в параметричній копії для забезпечення випробувань ПЗ АСКП [14].

СПИСОК ЛІТЕРАТУРИ

1. Райчев І.Е., Харченко О.Г. Проблеми оцінювання якості критичних програмних систем при їх сертифікації // Проблемы программирования. – 2004. – № 2–3. – С.198–207.
2. Малежик А.И. Основы компьютерных технологий оперативного контроля полетов воздушных судов по полетной информации. – К.: КМУГА, 1996. – 124 с.
3. Райчев И.Э., Харченко А.Г., Яцков Н.А. Методы создания тестовых наборов данных при сертификационных испытаниях комплексов программ контроля полетов // Вісник НАУ. – 2001. – № 1. – С. 126–132.
4. Вельбицкий И.В. Технология программирования. – К.: Техника, 1984. – 279 с.
5. Шалыто А.А. Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // “Известия Академии наук. Теория и системы управления”. – 2000. – № 6. – С. 63–81.
6. Райчев И.Э., Харченко А.Г. Применение конечных автоматов для реализации алгоритмов контроля полетов воздушных судов // Вісник НАУ. – 2001. – № 3. – С. 136–140.
7. Райчев І.Е. Синтез автоматних моделей контролю // Вісник НАУ. – 2002. – № 2. – С. 43–52.
8. Глушков В.М. Синтез цифровых автоматов. – М.: Физматгиз, 1962. – 476 с.
9. Кобринский Н.Е., Трахтенброт Б.А. Введение в теорию конечных автоматов. – М.: Физматгиз, 1962. – 404 с.
10. Богатырев Р. Об автоматном и асинхронном программировании // Открытые системы. – 2001. – № 3.
11. Кузнецов Б.П. Психология автоматного программирования // BYTE/Россия. – 2000. – № 11. – С. 22–29.
12. Райчев І.Е. Технологія визначення показників вірогідності одновимірних та багатовимірних об'єктів контролю за інформацією параметричних бортових реєстраторів // Вісник НАУ. – 2002. – № 1. – С. 17–24.
13. Райчев І.Е. Технологія оцінювання характеристик якості програмного забезпечення автоматизованих систем контролю при сертифікаційних випробуваннях: Автореф. дис… канд. техн. наук: 05.13.06 / Національний авіаційний університет. – Київ, 2005. – 20 с.
14. Райчев І.Е., Харченко О.Г. Система автоматизації сертифікаційних випробувань програмного забезпечення контролю польотів // Вісник Черкаського державного технологічного університету. – 2003. – № 3. – С. 24–30.